

Linux based 3G Multimedia Mobile-phone

API Specification

[Telephony Service]

Draft 1.0

NEC Corporation

Panasonic Mobile Communication Ltd.

Contents

Preface	16
1. Packet Communication Watching Functions	17
1.1 Start watching packet communication	17
1.2 Stop watching packet communication.....	20
1.3 Get packet communication status	21
1.4 Request transmission of packet communication	23
1.5 Respond to transmission of packet communication	24
1.6 Request disconnection of packet communication	25
1.7 Specify APN settings.....	27
1.8 Reference APN settings	28
1.9 Specify target destination.....	31
1.10 Reference target destination	33
1.11 Request to forcibly disconnect packet communication	34
1.12 Get U-Plane connection status	35
1.13 Request temporary change of APN settings	36
1.14 Request to resume temporary change of APN settings	37
1.15 Get DNS record name	38
1.16 Get default APN selection.....	39
1.17 Request to get WPCG IP address	40
1.18 Request to forcibly disconnect packet communication	41
1.19 Request to initialize APN	42
1.20 Start watching external equipment for packet communication	44
1.21 Stop watching external equipment for packet communication.....	46
2. Voice Communication Service Functions	47
2.1 Starting voice communication monitoring.....	47
2.2 Ending voice Communication monitoring.....	51
2.3 obtaining voice communication status	52
2.4 Referring to connection destination information.....	54
2.5 Obtaining the length of the current call	58
2.6 Off-hook request.....	59
2.7 Disconnect request.....	60
2.8 Dial	62
2.9 Dial completion	63
2.10 Response to an incoing call.....	65
2.11 Forwarding an incoming call	67
2.12 Phone-answering machine.....	68
2.13 Response hold.....	69
2.14 Rejection an incoming call	70
2.15 Processing of operating multiple call	71
2.16 Requesting on-hook transmission	72
2.17 Processing of obtaining the call nmber	73
2.18 Starting DCF message monitoring.....	75
2.19 Ending DCF message monitoring	80
2.20 Processing of notifying start/stop a voice message	81

2.21 Start processing of sounding a hold tone during a call	82
2.22 Stop processing of sound ing a hold tone during a call	83
2.23 Processing of referring to the 64K communication/AV communication status	84
2.24 Processing of referring to the internal/external AV communication status	85
2.25 Processing of reading the communication status(with incoming type)	86
2.26 Starting line status monitoring.....	88
2.27 Ending line status monitoring.....	92
2.28 Obtaining the receive level	93
2.29 Obtaining line status information.....	94
2.30 Obtaining the within- or out-of-communication area status	96
2.31 PLMN Number information obtainment function.....	97
2.32 Obtaining information on stored answering messages	98
2.33 Setting the number of answering messages to be stored	99
2.34 N + 1 Obtaining selection of the incoming function – getting of the incoming function.....	100
2.35 N + 1 Setting selection of the incoming function.....	101
2.36 Registering additional service setting.....	102
2.37 Obtaining additional service setting	103
2.38 Deleting additional service setting.....	104
2.39 Deleting all additional service setting	105
2.40 Registering response message setting	106
2.41 Registering response message setting	107
2.42 Deleting response message setting	108
2.43 Deleting all response message setting	109
2.44 Requesting reconnection tone setting	110
2.45 Obtaining reconnection tone setting	110
2.46 Processing of obtaining the noise canceller setting status.....	112
2.47 Processing of noise canceller setting.....	112
2.48 Processing of noise canceller setting.....	114
2.49 Processing of call quality alarm setting.....	114
2.50 Processing of obtaining noise canceller permission/non-permission	116
2.51 Processing of referring to the radio unit status.....	116
2.52 Processing of obtaining the priority communication mode setting state	118
2.53 Processing of priority communication mode setting.....	118
2.54 Obtainig voice message sound setting	120
2.55 Voice message sound setting.....	120
2.56 Processing of obtaining the setting status of automatic incoming ON/OFF	122
2.57 Automatic incoming ON/OFF setting processing	122
2.58 Processing of obtaining the setting status of the automatic incoming timer value	124
2.59 Processing of automatic incoming timer value setting.....	124
2.60 Obtaining the totalized reset data	126
2.61 Setting the totalized reset date	127
2.62 Setting inter-application transmittance event monitoring	128
2.63 Ending inter-application transmittance event monitorting.....	130
2.64 Requesting a notification of an inter-application transmittance event.....	131
2.65 Processing of call time registration	134
2.66 Processing of obtaining calling operation start time.....	135

2.67 Processing of setting of calling operation starttime	135
2.68 Obtaining absence incoming call history display setting	136
2.69 Processing of absence incoming call history deisplay setting	136
3. SMS Service	138
3.1 SMS Communication Functions Details	138
3.2 stop SMS communication monitoring	140
3.3 stop SMS communication monitoring	141
3.4 SMS-MT receive response	146
3.5 SMS-MT receive failure response	149
3.6 SMS receive memory check	151
3.7 stop SMS communication monitoring	152
3.8 Get SMS communication status	153
4. Equipment Service	154
4.1 Equipment Service event notice request start	154
4.2 Equipment Service event notice request stop	156
4.3 Earphone mode setup	157
4.4 Manner mode ON/OFF	158
4.5 Manner mode serup	160
4.6 Original manner mode serup	161
4.7 Vibrator operation start/stop	164
4.8 Vibrator pattern setup	166
4.9 Monitor mode start/stop	168
4.10 Secret data type setup	169
4.11 Terminal lock (all-feature lock) ON/OFF	170
4.12 PIM loc ON/OFF	171
4.13 Keypad-activated dialing ban ON/OFF	172
4.14 Charge start/end confirmatory sound setup	173
4.15 Rear LCD sleep mode ON/OFF	174
4.16 Battery level reference	175
4.17 Earphone connection status reference	176
4.18 Earphone mode reference	177
4.19 Open/closed status reference	178
4.20 Charger status reference	179
4.21 Manner mode ON/OFF reference	180
4.22 Manner mode setup reference	181
4.23 Vibration pattern name reference	182
4.24 Power ON status reference	184
4.25 Secret mode reference	185
4.26 Vibration pattern reference	186
4.27 Manner mode operation status reference	188
4.28 Terminal lock(all feature lock) ON/OFF status reference	189
4.29 PIM lock ON/OFF status reference	190
4.30 Keyboard-activated dialing ban ON/OFF reference	191
4.31 Battery pack detached-attached status reference	192
4.32 Operation mode acquisition processing	193
4.33 IMEI reference	194
4.34 USB connection status reference	195

4.35 UIM insertion status reference	196
4.36 Charge start/end confirmatory sound status reference.....	197
4.37 Vibrator operation status reference	198
4.38 Banned-operation setup status reference	199
4.39 Equipment Service message reference.....	201
4.40 User-specified item reset	202
4.41 Forced power-OFF notice.....	203
4.42 Personal data setup/reference processing.....	204
4.43 Personal picture data setup/reference processing	207
5. Schedule Service	209
5.1 Starat requesting Service event notifications.....	209
5.2 Stop requesting Schedule Service event notifications.....	211
5.3 register schedule event	212
5.4 Check on identical date/time setting for schedule data	214
5.5 Delete all schedules designated before specified date/time	217
5.6 Read schedule data by category	219
5.7 Read the number of registered schedules	221
5.8 Read the number of schedules registered to specified date	222
5.9 Search for schedule not alarmed.....	224
5.10 Read number of days in specified year/month.....	226
5.11 Read day of week of specified date	228
5.12 Read valid date.....	230
5.13 Register schedule-related setting status.....	232
5.14 Read schedule-related setting status	234
5.15 posting alarm notifications.....	236
5.16 Register user's icon	237
5.17 Read user's icon.....	239
5.18 Check for original data in use	241
5.19 Delete multiple schedules.....	242
5.20 Set system date/time.....	244
5.21 Read system date/time.....	246
5.22 Convert specified date and time to seconds	248
5.23 Convert seconds to date and time	250
5.24 Check schedule data	252
5.25 Convert seconds to date and time (GMT time zone).....	254
5.26 Register wake-up alarm.....	256
5.27 Delete wake-up alarm.....	258
5.28 Read wake-up alarm	259
5.29 Register one ToDo data item	261
5.30 Delete one ToDo data item.....	263
5.31 Delete ToDo data matching specified conditions	264
5.32 Delete all ToDo data.....	265
5.33 Read one ToDo data item.....	266
5.34 Read one ToDo data item by category	267
5.35 Get number of registered ToDo data items	269
5.36 Set ToDo data status.....	270
5.37 Set number of ToDo data items by category and status	272

5.38 Search for ToDo data not alarmed.....	274
5.39 Delete multiple ToDo data items.....	275
5.40 Register Java schedule event	277
5.41 Delete Java schedule event	279
5.42 Delete all Java schedule events.....	280
5.43 Read Java schedule event.....	281
5.44 Register Java schedule event	282
5.45 Delete Java schedule event	286
5.46 Delete all Java schedule events.....	287
5.47 Read Java schedule event.....	288
5.48 Read inactivated Java schedules from list.....	289
5.49 Clear list of inactivated Java schedules.....	291
5.50 Register holiday	292
5.51 Delete holiday.....	294
5.52 Delete all holidays.....	296
5.53 Delete all holidays before specified date	297
5.54 Read holiday.....	298
5.55 Read holiday list.....	301
5.56 Read number of registered holidays.....	303
5.57 Check another holiday on identical date.....	304
5.58 Read number of holidays registered before specified date	306
5.59 Reset default holidays.....	308
5.60 Get default holidays (in specified year/month)	309
5.61 Register anniversary.....	311
5.62 Delete anniversary.....	313
5.63 Delete all anniversaries.....	314
5.64 Delete all anniversaries before specified date	315
5.65 Read anniversary	317
5.66 Read anniversary list.....	319
5.67 Read number of registered anniversaries.....	321
5.68 Check another anniversary on identical date.....	322
5.69 Read number of anniversaries before specified date	324
5.70 Check for holiday/anniversary	326
5.71 Check for holiday/anniversary (in specified year/month).....	328
5.72 Register ADL reserved time event	330
5.73 Delete ADL reserved time event.....	332
5.74 Read ADL reserved time event.....	333
5.75 Read ADL re-calculated time.....	335
6. Data Exchange Service	337
6.1 Start Event notification request.....	337
6.2 Stop Event notification request.....	340
6.3 Language List Collection	341
6.4 Memory Language Change Request	342
6.5 USIM Language Setting Request.....	344
6.6 Get current language request.....	346
6.7 Get Self Phone Number (MSISDN) Request	347
6.8 Time/charges Phone Call Reset Request.....	349

6.9 Regular Reception Start Request	351
6.10 USIM version number request	352
6.11 Conversation/Total Time Request.....	353
6.12 Memory Dial Read	354
6.13 Memory Dial Additional Information Request	359
6.14 Memory Dial Additional Information Deletion.....	362
6.15 Memory Dial Registration (Main Part)	364
6.16 Memory Dial Registration (UIM)	366
6.17 Memory Dial(Main Part)	368
6.18 Memory Dial Deletion (UIM).....	369
6.19 Memory Dial Registration Status Request	371
6.20 Sort Table Creation.....	373
6.21 Sort Table Read.....	378
6.22 Memory Dial Keyword Search.....	380
6.23 Memory Number Check.....	383
6.24 Memory Number Request.....	385
6.25 Memory Dial Attribute Setting/Release.....	387
6.26 Search Object Count Request.....	389
6.27 Sort Table Single Data Item Deletion	391
6.28 Sort Table Search.....	393
6.29 Priority Telephone Number Search.....	394
6.30 Priority Mail Address Search	396
6.31 Set/Release Specified Function Request	397
6.32 Count Specified Function Request	401
6.33 Enable/Disable Specified function Request	402
6.34 Specified Function Status Check.....	404
6.35 Initialize Specified Function Information.....	407
6.36 Clear Specified Function Condition	408
6.37 Request Specified Function Setting Enabled/Disabled	410
6.38 Check Specified Incoming Image Setting	412
6.39 Change Specified Incoming Image	414
6.40 Create Specified Function Sort Table	416
6.41 Read Specified Function Sort Table	418
6.42 Count Specified Function Object From Search Table.....	420
6.43 Delete Single Data Item From Specified Function Sort Table.....	422
6.44 Request USIM Telephone Directory Structure Information	424
6.45 Telephone Directory Group Name Request.....	426
6.46 Set Telephone Directory Group Name Request (Main Part).....	427
6.47 Set Telephone Directory Group Name Request (USIM).....	428
6.48 Copy USIM Telephone Directory Memory Request.....	430
6.49 Clear USIM Cache Data Request.....	432
6.50 Check Telephone Directory Registration Data	433
6.51 Check Specified Ring Tone Setting.....	435
6.52 Change Specified Ring Tone	436
6.53 Auto Display Status Request.....	437
6.54 Memory Dial Total Deletion	438
6.55 SMS Data Add/Update.....	439

6.56 SMS Data Read	443
6.57 SMS Data Deletion	445
6.58 SMS Parameter Setting.....	447
6.59 Get SMS Parameter Request.....	450
6.60 SMS Status Report Deletion.....	452
6.61 SMS Memory Capacity Flag Setting.....	453
6.62 Get SMS Memory Capacity Flag Request.....	455
6.63 Send APDU Request	457
6.64 PUBKEY Read Request.....	460
6.65 Digital Signature Operation Execution Request	462
6.66 UIM Certificate Read Request	465
6.67 UIM Certificate Enable/Disable Setting Request.....	467
6.68 UIM Cache Disable Request.....	469
6.69 UIM Certificate Cache Data Access	470
6.70 History Registration	472
6.71 History Access Start.....	477
6.72 History Access	480
6.73 History Deletion.....	485
6.74 Delete Unconfirmed Absent Incoming Flag Request.....	486
6.75 History Access Stop.....	487
6.76 Total Deletion by History Type.....	488
6.77 History Count Reques.....	489
6.78 Unconfirmed Absent Incoming History Count Request	490
6.79 Address History Setting Access	491
6.80 Address History Deletion.....	493
6.81 Address History Total Deletion	494
6.82 Address History Registration Count Request.....	495
7. Record/Playback.....	496
7.1 Record/Playback Event Notification Request Stop.....	496
7.2 Record Data information Collection Request	497
7.3 Voice Data Count Information Collection	501
7.4 Voice Message Memo Start.....	503
7.5 Verbal Message Memo Start Access	505
7.6 Record/Playback Status Access	507
7.7 Next Playback Data Information Collection.....	512
7.8 Verval Message Memo Path Setting	514
7.9 Verval Message Memo Message Pattern Setting.....	516
7.10 Verval Message Memo Message Access.....	518
7.11 Message Title (Response On-Hold Tone and Concersation On-Hand Tone).....	520
7.12 Response On Hold Tone Setting	524
7.13 Response On Hold Tone Access.....	526
7.14 Downward playback Check	528
7.15 TV Verbal Message Memo Response Message Status Registration	529
7.16 TV Verbal Message Memo Response Message Status Access	530
7.17 Conversation On-Hold Tone Setting	531
7.18 Conversation On-Hold Tone Access.....	533
7.19 Record Start Request.....	535

7.20 Playback Start Request	542
7.21 Record/Playback Stop Request	551
7.22 Record Data Deletion Request.....	553
7.23 Record Start Request (Confirmation Tone Flag Attached).....	556
7.24 Playback Start Request(Confirmation Tone Flag Attached).....	558
7.25 Record/Playback Stop Request (Confirmation Tone Flag Attached).....	560
8. LMP MANAGEMENT SERVICE.....	562
8.1 Request to Turn Backlight On.....	562
8.2 Request to Turn Backlight Off.....	566
8.3 Request to Turn LED On	569
8.4 Request to Turn LED Off.....	574
8.5 Referencing Backlight Lighting state	577
8.6 Referencing LED Lighting State	578
8.7 Listing LMP Titles and Collecting Color Numbers	580
8.8 Referncing Original LMP Colors	583
8.9 Setting Original LMP Colors.....	586
8.10 Changing Original LMP Color Titles	589
8.11 Referencing LMP Setting.....	591
8.12 Referencing LMP Settings.....	594
8.13 Referencing LMP Incoming Illumination Settings.....	597
8.14 Referencing LMP Incoming Illumination Setting	599
8.15 Request to Turn Rear Emblem On	601
8.16 Request to Turn Rear Emblem Off.....	604
8.17 Referencing incoming LED-associated Settings	606
8.18 Registering incoming LED-associated Settings.....	607
8.19 Notifying of Change in Operation Modes.....	608
8.20 Notifying of Change in Operation Modes.....	611
8.21 Registering Communicating Illumination Settings	613
8.22 Registering Turn Rear Backlight On	615
8.23 Registering Turn Rear Backlight Off	620
9. Sound System.....	622
9.1 Sound System Event Occurrence Notification Request	622
9.2 Sound System Event Occurrence Notification Release	623
9.3 Original Ring Tone Usage Status Collection	624
9.4 Original Ring Tone Unused Area Information Collection	626
9.5 Original Ring Tone Unused Area Information Collection	627
9.6 Original Ring Tone Information Modification	628
9.7 Original Ring Tone Data Collection	631
9.8 Original Ring Tone Information Collection.....	633
9.9 Melody player list registration	635
9.10 Original Ring Tone Registration	637
9.11 Original Ring Tone Deletion	639
9.12 Melody Sort Setting	641
9.13 Melody Sort Access.....	643
9.14 User-Defined Folder List Collection.....	644
9.15 Folder Melody Usage Status Collection	646
9.16 User-Defined Folder Creation Editing	648

9.18 Melody Data Folder Contents Move.....	652
9.19 Original Ring Tone Registration Folder Collection	654
9.20 Ring Tone Volume Setting	655
9.21 Ring Tone Volume Access.....	657
9.22 Talk Volume Setting.....	659
9.23 Talk volume Access	660
9.24 Built-in Hands Free On/Off Access	661
9.25 Built-in Hands Free On/Off Setting.....	662
9.26 Microphone Amplifier Gain Switching.....	663
9.27 Path Setting	664
9.28 Ring Tone Setting.....	665
9.29 Ring Tone Access	667
9.30 Ring Tone Sound Time Setting.....	669
9.31 Ring Tone Sound Time Access	670
9.32 Key Sound On/Off Setting	671
9.33 Key Sound On/Off Access.....	672
9.34 Default Sound ID Collection.....	673
9.35 Operation Mode Setting.....	674
9.36 DSP Power Supply Control.....	675
9.37 Mute Control	676
9.38 Movie Sound Record/Playback Status Settingl	677
9.39 DSP Parameter Modification.....	678
9.40 Path Access.....	679
9.41 Media Sound Start.....	680
9.42 Media Sound Stop	683
9.43 Sound Start	685
9.44 Sound Stop	692
9.45 Videophone File Playback Start	696
9.46 Videophone File Playback Stop	698
9.47 Videophone DTMF Start.....	699
10. SD File.....	700
10.1 SD management file service event registration	700
10.2 SD management file service event release.....	704
10.3 Processing halt (asynchronous)	706
10.4 File read.....	707
10.5 File registration.....	709
10.6 File deletion	712
10.7 New file information open.....	714
10.8 New file information close.....	718
10.9 Save destination folder set.....	720
10.10 File read (asynchronous).....	722
10.11 File registration (asynchronous)	724
10.12 File deletion (asynchronous)	727
10.13 Delete all files in folder	728
10.14 Folder registration (asynchronous).....	731
10.15 Folder deletion (asynchronous).....	733
10.16 File copy (asynchronous)	735

10.17	File movement (asynchronous)	736
10.18	Title change (asynchronous)	738
10.19	Single vObject export (asynchronous)	740
10.20	All vObject items export (asynchronous)	742
10.21	Single vObject import (asynchronous)	744
10.22	All vObject items import (asynchronous)	746
10.23	Folder initialization (asynchronous)	748
10.24	Total folder/file list count acquisition	750
10.25	Folder/file list information acquisition	753
10.26	Total free area size acquisition	757
10.27	SD card information acquisition	758
10.28	SD use size acquisition function	760
10.29	File detailed information acquisition	761
10.30	Parent folder information acquisition	764
10.31	File presence check	767
10.32	Information for indicator acquiring function	769
10.33	Card size information acquisition	770
10.34	Save destination folder data ID acquisition	771
10.35	SD card insertion status acquisition	772
10.36	File detailed information acquisition (asynchronous)	773
10.37	Folder/file list information acquisition (asynchronous)	775
10.38	vObject list information acquisition	778
10.39	Detailed vObject information acquisition	781
10.40	Check disk	783
10.41	SD card format (asynchronous)	784
10.42	Check disk (asynchronous)	786
10.43	DPOF print target file count acquisition (synchronous)	788
10.44	DPOF print target page count acquisition (asynchronous)	789
10.45	DPOF print target page count set (asynchronous)	791
10.46	DPOF target file deletion (asynchronous)	793
10.47	File open	795
10.48	File closing	797
10.49	File read	798
10.50	File writing	800
10.51	Seek	801
10.52	Total free area size (byte) acquisition	801
10.53	File deletion	803
10.54	File information acquisition	804
10.55	SD general area format check	805
10.56	File error acquisition	806
11.	Resource Management Library Interface	807
11.1	Resetting User-defined Items	807
11.2	Setting or Referencing Communication Parameters	808
11.3	Setting or Referencing Functions	810
11.4	Setting or Referencing Tone Gain Specification (CDTNG or CDTONE) 2	811
11.5	Setting or Referencing Wakeup Message	814
11.6	Setting or Referencing Clock Display	815

11.7	Setting or Referencing Formatted Text Folder Name.....	817
11.8	Setting Original Menu Number	818
11.9	Setting or Referencing Image Paste Destination Specification.....	819
11.10	Setting or Referencing Pose Dial	822
11.11	Setting or Referencing Original Manner Mode	823
11.12	Setting or Referencing Various Ring Tone Patterns	825
11.13	Registering or Referencing Fixed Formatted Texts	827
11.14	Deleting Free Formatted Texts (Range Specification)	829
11.15	Registering or Referencing Free Formatted Texts	830
11.16	Setting or Referencing Compact Desktop Information 2	832
11.17	Deleting Compact Desktop 2.....	836
11.18	Setting or Referencing Image Position	837
11.19	Setting or Referencing APN Data 2	839
11.20	Setting or Referencing Color Name or RGB Information	841
11.21	Setting or Referencing Order of Original Animations 2	843
11.22	Setting or Referencing Melody Player Program List	844
11.23	Deleting Melody Player Program List	845
11.24	Deleting All Items on Melody Player Program List	846
11.25	Operating (Setting, Referencing, or Deleting) Mailing List Name	847
11.26	Operating (Setting, Referencing, or Deleting) Mail Address	849
11.27	Referencing Whether Mail Address Is Registered	851
11.28	Referencing (Listing) Mail Address	853
11.29	Referencing Number of Mail Addresses Registered.....	855
11.30	Referencing Total Number of Mail Addresses Registered	857
11.31	Setting or Referencing Alarm Suspension Information 2	858
11.32	Setting or Referencing SMScenter (User-defined)	860
11.33	Setting or Referencing Certificate Download.....	862
11.34	Setting or Referencing Various Ring Tone Patterns for Using TV Phone.....	864
11.35	Setting or Referencing Communication Mode.....	865
11.36	Setting or Referencing Various TV Phone Information Items.....	866
11.37	Setting or Referencing Remote Monitoring	869
11.38	Setting or Referencing Continuous Image Titles	870
11.39	Setting or Referencing Order of Continuous Images	871
11.40	Deleting Continuous Images.....	872
11.41	Setting or Referencing Recording Image Quality	873
11.42	Setting or Referencing Recording Size Selection	874
11.43	Setting or Referencing Continuous Shooting Function	875
11.44	Setting or Referencing Brightness Adjustment.....	876
11.45	Setting or Referencing UIM Information	877
11.46	Setting or Referencing EEPROM Area.....	879
11.47	Setting or Referencing Self-produced Animation Title	880
11.48	Setting Voice Clock	881
11.49	Setting or Referencing Animation Backlight	882
11.50	Setting or Referencing Communication Time	883
11.51	Initializing Communication Time or Calling Rate.....	885
11.52	Setting or Referencing Voice or Message Slip Management Information	887
11.53	Registering or Referencing Service Setting Number	889

11.54	Deleting Service Setting Number	891
11.55	Setting or Referencing IMEI	892
11.56	Setting or Referencing Accumulated Reset Date and Time	893
11.57	Setting or Referencing Response Message	894
11.58	Deleting Response Message	896
11.59	Setting or Referencing Camera Shooting Image Size	897
11.60	Setting or Referencing White Balance	899
11.61	Setting or Referencing Various Setting Information Items about Camera Function 2	900
11.62	Setting or Referencing MPEG-4 Encoder Parameters 2	902
11.63	Setting or Referencing Encoder parameters for Recording Animations	910
11.64	Setting or Recording Replay Animation Display Method	913
11.65	Setting or Referencing Prefix	914
11.66	Setting or Referencing Camera or Download Folder Name	916
11.67	Setting or Referencing MPEG4 Capability	917
11.68	Setting or Referencing Terminal Manufacturer	918
11.69	Setting or Referencing H.223 Level	919
11.70	Setting or Referencing Reciprocating Delay Time	920
11.71	Setting or Referencing Video Encoding Parameters	921
11.72	Setting or Referencing QOS Parameters	923
11.73	Setting or Referencing Image Display Method	925
11.74	Setting or Referencing Individual Information about Communication Data	926
11.75	Registering SD Power-off (SDM)	928
11.76	Setting or Referencing SD-PIM Recovery Information	929
11.77	Setting or Referencing Antenna Bar Level	931
11.78	Setting or Referencing Profiles	932
11.79	Registering or Referencing Serial Production Number	936
11.80	Referencing Nonvolatile Area (EEPROM) Initial Value	938
11.81	Referencing Nonvolatile Area (EEPROM)	939
11.82	Setting Nonvolatile Area (EEPROM)	940
11.83	Setting or Referencing Priority Connection Destination Setting Data	941
11.84	Recovery Request after Power-off	943
11.85	Collecting Display Color	944
11.86	Setting or Collecting Color Pattern Number	946
11.87	Operating Text Memo	948
11.88	Deleting Text Memo	951
11.89	Referencing Text Memo List	952
11.90	Collecting Number of Text Memo Data Items	954
11.91	Text Memo Operation	955
11.92	Wording Collection Processing	958
11.93	Wording Size Collection Processing	959
11.94	Request to Collect Image Display Data	960
11.95	Request to Register Image Display Data	967
11.96	Request to Erase Image Display Data	970
11.97	Request to Write Image Display Data Title	973
11.98	Request to Collect List of Image Display Data Headers	974
11.99	Request to Collect Image Display Data Header Information	978

11.100	Request to Rename Image Display Data File	980
11.101	Request to Collect Number of Image Display Data Items Registered	981
11.102	Request to Collect Information about Image Display Data Paste Destination	983
11.103	Request to Set Information about Image Display Data Paste Destination	988
11.104	Request to Cancel Information about Image Display Data Paste Destination	990
11.105	Setting or Referencing Image Display Data Paste Information (Cutout or Display Position)	992
11.106	Setting or Referencing User-defined Animation Information	994
11.107	Checking User-defined Animation Utilization Status	996
11.108	Request to Create Image Display Data Folder	998
11.109	Request to Erase Image Display Data Folder	1000
11.110	Request to Move Image Display Data Folder	1002
11.111	Request to Collect Number of Image Display Data Folders	1004
11.112	Request to Collect Image Display Data Folder Name.....	1006
11.113	Request to Collect Name of Folder that Registers Image Display Data	1008
11.114	Setting or Referencing Display Switching Information	1010
11.115	Request to Collect Image Folder Image ID.....	1012
11.116	Request to Delete Multiple Image Display Data Items	1014
11.117	Request to Collect Icon Information	1016
11.118	Request to Collect Icon Image	1018

Preface

This document describes an API specification of Telephony Service for Linux based 3G multimedia mobile-phone. This document is the results of the work of the CE Linux Forum's technical working group. The APIs in this document are based on the technology which is originally the collaborative work by NEC Corporation, Panasonic Mobile Communication Ltd., and NTT DoCoMo, Inc.

1. Packet Communication Watching Functions

1.1 Start watching packet communication

1-1 Start watching packet communication			
Classification	PS Service/Packet communication watching		
Function	Start watching packet communication		
Symbol	Elib_CP_Start_Com_Watch		
Syntax	int Elib_CP_Start_Com_Watch (unsigned int ap_id, int mask, MsbFunc callback_func) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Mask	Int	I	<p>Notification event mask Enabled with the bit turned ON</p> <p>ELIB_CP_MASK_DCF_MSG 0x01: DCF message notification (exclusive of data transmission/reception)</p> <p>ELIB_CP_MASK_TAF_MSG 0x02: PS-TAF message notification</p> <p>ELIB_CP_MASK_DCF_DAT 0x04: DCF message notification (for data transmission/reception)</p> <p>ELIB_CP_MASK_ALL 0xff: All (*1)</p> <p>*1: To reduce the load of message communication, make a request only for a required notification event. When notice of the event is no longer needed, stop giving notice by using "A-2 Stop watching packet communication."</p>
callback_func	MsbFunc	I	Callback function of which the event is notified
Return value	Type	I/O	Description
Sts	int	O	<p>Result of processing</p> <p>ELIB_CP_OK: Successful</p> <p>ELIB_CP_NG: Unsuccessful</p>
Include File	target/v4t_le/include/srv_cp.h		
Functional description	<p>This function starts giving notice about events related to packet communication.</p> <p>This function notifies a requesting application of the following events:</p> <p><Notification events></p> <p><u>DCF message notification:</u></p> <ul style="list-style-type: none"> - Gives notice of messages related to packet communication (exclusive of data transmission/reception) from DCF. <p><u>PS-TAF message notification:</u></p> <ul style="list-style-type: none"> - Gives notice of messages from PS-TAF. 		

DCF message notification (for data transmission/reception):

- Gives notice of messages related only to packet communication (data transmission/reception) from DCF.
- Notification of data transmitting indication started
- Notification of data receiving indication started

<DCF message notification event>

```
typedef struct {
    int          category ;           /* PacketNotify */
    int          subtype ;           /* PacketNotify_DCF */
    int          info ;              /* DCF message type (packet
communication status) */
    int          subinfo ;           /* PDP type */
    union {
        ...
        DCF message structure ;/* DCF message */
        ...
    } data ;
} _ELIB_CP_EVENT ;
```

*** PDP type**

```
ELIB_CP_PDP_TYPE_X25      : X.25
ELIB_CP_PDP_TYPE_IP       : IP
ELIB_CP_PDP_TYPE_IPV6     : IPV6
ELIB_CP_PDP_TYPE_OSPIH    : OSPIH
ELIB_CP_PDP_TYPE_PPP      : PPP (IP based PPP)
```

<PS-TAF message notification event>

```
typedef struct {
    int          category ;           /* PacketNotify */
    int          subtype ;           /* PacketNotify_TAF */
    int          info ;              /* PS-TAF message type */
    int          subinfo ;
    union {
        ...
        PS-TAF message structure ; /* PS-TAF message */
        ...
    } data ;
} _ELIB_CP_EVENT ;
```

<DCF message (data transmission/reception) notification event>

```
typedef struct {
    int          category ;           /* PacketNotify */
    int          subtype ;           /* PacketNotify_DCF_DAT */
    int          info ;              /* DCF message type (for data
transmission/reception) */
```

	<pre> int subinfo ; /* PDP type */ union { ... DCF message structure ; /* DCF message */ ... } data ; } _ELIB_CP_EVENT ; * PDP type ELIB_CP_PDP_TYPE_X25 : X.25 ELIB_CP_PDP_TYPE_I : IP ELIB_CP_PDP_TYPE_IPV6 : IPV6 ELIB_CP_PDP_TYPE_OSPIH : OSPIH ELIB_CP_PDP_TYPE_PPP : PPP(IP based PPP) </pre>
Related message	Messages from PS-TAF/DCF
Necessary procedure	None
Note	None
Prohibition	None
Use example	None

1.2 Stop watching packet communication

1-2 Stop watching packet communication			
Classification	PS Service/Packet communication watching		
Function	Stop watching packet communication		
Symbol	Elib_CP_Stop_Com_Watch		
Syntax	int Elib_CP_Stop_Com_Watch (unsigned int ap_id, int mask) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Mask	int	I	Notification event mask Stops a message notification whose bit turned ON ELIB_CP_MASK_DCF_MSG 0x01: DCF message notification ELIB_CP_MASK_TAF_MSG 0x02: PS-TAF message notification ELIB_CP_MASK_DCF_DAT 0x04: DCF message notification (for data transmission/reception) ELIB_CP_MASK_ALL 0xff: All
Return value	Type	I/O	Description
Sts	int	O	Result of processing ELIB_CP_OK: Successful ELIB_CP_NG: Unsuccessful
Include File	target/v4t_le/include/srv_cp.h		
Functional description	This function stops giving notice of specified events about packet communication. For details about notification events, see "A-1 Start watching packet communication."		
Related message	None		
Necessary procedure	A request for notification events has been made by using the "A-1 Start watching packet communication" in advance.		
Note	None		
Prohibition	None		
Use example	None		

1.3 Get packet communication status

1-3 Get packet communication status			
Classification	PS Service/Packet communication watching		
Function	Get packet communication status		
Symbol	Elib_CP_Get_Com_Status		
Syntax	int Elib_CP_Get_Com_Status (unsigned int ap_id, int pdp_type);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
pdp_type	int	I	Reference PDP type ELIB_CP_PDP_TYPE_X25 : X.25 ELIB_CP_PDP_TYPE_IP : IP ELIB_CP_PDP_TYPE_IPV6 : IPV6 ELIB_CP_PDP_TYPE_OSPIH : OSPIH ELIB_CP_PDP_TYPE_PPP : PPP (IP based PPP)
Return value	Type	I/O	Description
Sts	int	O	Packet communication status ELIB_CP_COM_STATUS_EMPTY : Communication ended (idle) ELIB_CP_COM_STATUS_ACCEP_CONN: Connecting for reception ELIB_CP_COM_STATUS_CONNECT : Connecting for transmission ELIB_CP_COM_STATUS_ACTIVE : Communication active (C-Plane active) ELIB_CP_COM_STATUS_DISCONNECT: Disconnecting ELIB_CP_NG : Error
Include File	target/v4t_le/include/srv_cp.h		
Functional description	This function gets the current packet communication status of the specified PDP type. This function allows to get only the status without calling "A-1 Start watching packet communication."		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example	To reference the IP communication status and PPP communication status, make a separate call to this function for each of them (pdp_type does not allow the OR operator):		

	<pre>int ip_status = Elib_CP_Get_Com_Status(ap_id, ELIB_CP_PDP_TYPE_IP); int ppp_status = Elib_CP_Get_Com_Status(ap_id, ELIB_CP_PDP_TYPE_PPP);</pre>
--	--

1.4 Request transmission of packet communication

1-4 Request transmission of packet communication			
Classification	PS Service/Packet communication watching		
Function	Request transmission of packet communication		
Symbol	Elib_CP_Connect_Req		
Syntax	int Elib_CP_Connect_Req (unsigned int ap_id , int pdp_type) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
pdp_type	int	I	Transmission PDP type ELIB_CP_PDP_TYPE_IP : IP ELIB_CP_PDP_TYPE_PPP : PPP (IP based PPP)
Return value	Type	I/O	Description
Sts	int	O	Result of processing ELIB_CP_OK: Successful ELIB_CP_NG: Unsuccessful ELIB_CP_IMOD_DENIED: Transmission rejected due to no subscription to packet communication service
Include File	target/v4t_le/include/srv_cp.h		
Functional description	This function starts (originates) packet communication. - This function makes a request to PS-TAF for AT-BIN(+CGACT=1).		
Related message	Messages from PS-TAF/DCF For a return of ELIB_CP_IMOD_DENIED, PS Service gives notice of "A-4 Notification of packet transmission rejected (ELIB_CP_CALLING_REJ_IND)."		
Necessary procedure	To check the result, issue "A-1 Start watching packet communication," then get a message of DCF or PS-TAF.		
Note	None		
Prohibition	None		
Use example	None		

1.5 Respond to transmission of packet communication

1-5 Respond to transmission of packet communication			
Classification	PS Service/Packet communication watching		
Function	Respond to transmission of packet communication		
Symbol	Elib_CP_Connect_Rsp		
Syntax	int Elib_CP_Connect_Rsp (unsigned int ap_id , int type, unsigned char cr) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Type	int	I	Type ELIB_CP_COM_CONN_ACCEPT : Accepts termination ELIB_CP_COM_CONN_REJECT : Rejects termination
Cr	unsigned char	I	Call number Specifies the call number to be responded for termination.
Return value	Type	I/O	Description
Sts	int	O	Result of processing ELIB_CP_OK: Successful ELIB_CP_NG: Unsuccessful ELIB_CP_ANS_ERROR: Error in response for termination
Include File	target/v4t_le/include/srv_cp.h		
Functional description	<p>This function starts packet communication (accepts termination) or rejects it.</p> <ul style="list-style-type: none"> - For acceptance, this function makes a request to PS-TAF for AT-BIN(+CGANS=1). - For rejection, this function makes a request to PS-TAF for AT-BIN(+CGANS=0). <p>In the call number argument, specify the call number posted with a DCF/PS-TAF message.</p>		
Related message	Messages from PS-TAF/DCF		
Necessary procedure	To check the result, issue "A-1 Start watching packet communication," then get a message of DCF or PS-TAF.		
Note	None		
Prohibition	None		
Use example	None		

1.6 Request disconnection of packet communication

1-6 Request disconnection of packet communication			
Classification	PS Service/Packet communication watching		
Function	Request disconnection of packet communication		
Symbol	Elib_CP_Disconnect_Req		
Syntax	int Elib_CP_Disconnect_Req (unsigned int ap_id , unsigned char cr) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Cr	unsigned char	I	Call number Specifies the call number to be disconnected.
Return value	Type	I/O	Description
Sts	int	O	Result of processing ELIB_CP_OK: Successful ELIB_CP_NG: Unsuccessful
Include File	target/v4t_le/include/srv_cp.h		
Functional description	<p>This function closes (disconnects) packet communication.</p> <ul style="list-style-type: none"> - This function makes a request to PS-TAF for AT-BIN(+CGACT=0). <p>In the call number argument, specify the call number posted with a DCF/PS-TAF message.</p>		
Related message	Messages from PS-TAF/DCF		
Necessary procedure	To check the result, issue "A-1 Start watching packet communication," then get a message of DCF or PS-TAF.		
Note	None		
Prohibition	None		
Use example	<p>[Canceling an transmission] An transmission can be canceled with a procedure same as the regular procedure for disconnection. Set the call number included in the notification of transmitting indication started.</p> <pre>sts = Elib_CP_Disconnect_Req(ap_id, <DCF_PCP_CALLING_START_REQ>.cr) ;</pre> <p>* Although the call number may possibly be unacquired if an transmission is cancelled before the notification of transmitting indication started is received, a notification of transmitting indication started is posted whenever the transmission request SMREG-PDP-activate-req is sent to the network side, thus any transmission can never be cancelled before a notification of transmitting indication started is received, so the call number cannot be unacquired.</p>		



1.7 Specify APN settings

1-7 Specify APN settings			
Classification	PS Service/Packet communication watching		
Function	Specify APN settings		
Symbol	Elib_CP_APN_Set		
Syntax	<pre>int Elib_CP_APN_Set (unsigned int ap_id , int Mode, unsigned char *Apn, unsigned char *Host, unsigned char *Title) ;</pre>		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Mode	Int	I	Destination APN number ELIB_CP_APN_USERSET1: User-specified 1 ELIB_CP_APN_USERSET2: User-specified 2 ELIB_CP_APN_USERSET3: User-specified 3 ELIB_CP_APN_USERSET4: User-specified 4 ELIB_CP_APN_USERSET5: User-specified 5 ELIB_CP_APN_USERSET6: User-specified 6 ELIB_CP_APN_USERSET7: User-specified 7 ELIB_CP_APN_USERSET8: User-specified 8 ELIB_CP_APN_USERSET9: User-specified 9 ELIB_CP_APN_USERSET10: User-specified 10
Apn	unsigned char *	I	User-specified APN data - The allowable size is 1 to ELIB_CP_APN_MAX + NULL.
Host	unsigned char *	I	User-specified host data - The allowable size is 1 to ELIB_CP_HOST_MAX + NULL.
Title	unsigned char *	I	User-specified title data - The allowable size is 1 to ELIB_CP_TITLE_MAX + NULL.
Return value	Type	I/O	Description
Sts	Int	O	Result of processing ELIB_CP_OK: Successful ELIB_CP_NG: Unsuccessful
Include File	target/v4t_le/include/srv_cp.h		
Functional description	This function specifies APN settings. - For user-specified 1 to 10, set the user-specified APN data, user-specified host data, and user-specified title data acquired with the arguments.		

	ELIB_CP_APN_MAX /* Maximum number of characters for user-specified APN data */ ELIB_CP_HOST_MAX /* Maximum number of characters for user-specified host data */ ELIB_CP_TITLE_MAX /* Maximum number of characters for user-specified title data */ Note) The service side does not check any user-specified APN/host/title data, thus the APL side needs to check the APN data code.
Related message	None
Necessary procedure	None
Note	None
Prohibition	None
Use example	None

1.8 Reference APN settings

1-8 Reference APN settings			
Classification	PS Service/Packet communication watching		
Function	Reference APN settings		
Symbol	Elib_CP_APN_Ref		
Syntax	int Elib_CP_APN_Ref (unsigned int ap_id , _CPNotify_APN_HOST_DATA *apn_data);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
apn_data	_CPNotify_APN_HOST_DATA *	I/O	APN/host data structure; see *1.
Return value	Type	I/O	Description
Sts	Int	O	Result of processing ELIB_CP_OK: Successful ELIB_CP_NG: Unsuccessful
Include File	target/v4t_le/include/srv_cp.h		
Functional description	<p>This function references the APN/host data and title data of an APN number specified by parameters.</p> <p>This function references APN/host data of a specified APN number. To reference an APN number that is in use, use the "A-10 Reference target destination" function.</p> <p>Acquired from parameters: (APN/host data structure) *1</p> <ul style="list-style-type: none"> - APN data - Host data 		

	<p>- Title data</p> <p>ELIB_CP_APN_MAX /* Maximum number of characters for user-specified APN data */</p> <p>ELIB_CP_HOST_MAX /* Maximum number of characters for user-specified host data */</p> <p>ELIB_CP_TITLE_MAX /* Maximum number of characters for user-specified title data */</p> <p>*1)</p> <p>APN/host data structure:</p> <pre>typedef struct _CPNotify_APN_HOST_DATA { unsigned char apn_no ; /* [I/-] APN number */(*1) unsigned char APN[ELIB_CP_APN_MAX + 1] ; /* [-/O] APN data */ unsigned char HOST[ELIB_CP_HOST_MAX + 1] ; /* [-/O] Host data */ unsigned char TITLE[ELIB_CP_TITLE_MAX + 1] ; /* [-/O] Title data */ }_CPNotify_APN_HOST_DATA ;</pre> <p>(*1): For apn_no (APN number) in the APN/host data structure, specify any of the following APN numbers:</p> <p>APN number</p> <p>ELIB_CP_APN_DEFAULT : Default setting</p> <p>ELIB_CP_APN_USERSET1: User-specified 1</p> <p>ELIB_CP_APN_USERSET2: User-specified 2</p> <p>ELIB_CP_APN_USERSET3: User-specified 3</p> <p>ELIB_CP_APN_USERSET4: User-specified 4</p> <p>ELIB_CP_APN_USERSET5: User-specified 5</p> <p>ELIB_CP_APN_USERSET6: User-specified 6</p> <p>ELIB_CP_APN_USERSET7: User-specified 7</p> <p>ELIB_CP_APN_USERSET8: User-specified 8</p> <p>ELIB_CP_APN_USERSET9: User-specified 9</p> <p>ELIB_CP_APN_USERSET10: User-specified 10</p>
Related message	None
Necessary procedure	None
Note	None
Prohibition	None
Use example	None

1.9 Specify target destination

1-9 Specify target destination			
Classification	PS Service/Packet communication watching		
Function	Specify target destination		
Symbol	Elib_CP_APN_No_Set		
Syntax	int Elib_CP_APN_No_Set (unsigned int ap_id , int apn_no) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
apn_no	Int	I	Destination number to be used ELIB_CP_APN_DEFAULT : Default setting ELIB_CP_APN_USERSET1: User-specified 1 ELIB_CP_APN_USERSET2: User-specified 2 ELIB_CP_APN_USERSET3: User-specified 3 ELIB_CP_APN_USERSET4: User-specified 4 ELIB_CP_APN_USERSET5: User-specified 5 ELIB_CP_APN_USERSET6: User-specified 6 ELIB_CP_APN_USERSET7: User-specified 7 ELIB_CP_APN_USERSET8: User-specified 8 ELIB_CP_APN_USERSET9: User-specified 9 ELIB_CP_APN_USERSET10: User-specified 10
Return value	Type	I/O	Description
Sts	Int	O	Result of processing ELIB_CP_OK: Successful ELIB_CP_NG: Unsuccessful
Include File	target/v4t_le/include/srv_cp.h		

Functional description	<p>This function specifies a destination number to be used.</p> <p>This function only specifies a destination number to be connected.</p> <p>To specify APN/host data settings, use the "Specify APN settings" function.</p>
Related message	None
Necessary procedure	None
Note	None
Prohibition	None
Use example	

1.10 Reference target destination

1-10 Reference target destination			
Classification	PS Service/Packet communication watching		
Function	Reference target destination		
Symbol	Elib_CP_APN_No_Ref		
Syntax	int Elib_CP_APN_No_Ref (unsigned int ap_id) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Return value	Type	I/O	Description
apn_no	Int	O	Destination number to be used ELIB_CP_APN_DEFAULT : Default setting ELIB_CP_APN_USERSET1: User-specified 1 ELIB_CP_APN_USERSET2: User-specified 2 ELIB_CP_APN_USERSET3: User-specified 3 ELIB_CP_APN_USERSET4: User-specified 4 ELIB_CP_APN_USERSET5: User-specified 5 ELIB_CP_APN_USERSET6: User-specified 6 ELIB_CP_APN_USERSET7: User-specified 7 ELIB_CP_APN_USERSET8: User-specified 8 ELIB_CP_APN_USERSET9: User-specified 9 ELIB_CP_APN_USERSET10: User-specified 10
Include file	target/v4t_le/include/srv_cp.h		
Functional description	This function references the destination number which is being specified. This function only references a destination number to be connected. To specify APN/host data settings, use the " Reference APN settings " function.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example	None		

1.11 Request to forcibly disconnect packet communication

1-11 Request to forcibly disconnect packet communication			
Classification	PS Service/Packet communication watching		
Function	Request to forcibly disconnect packet communication		
Symbol	Elib_CP_Forced_Disc_Req		
Syntax	int Elib_CP_Forced_Disc_Req (unsigned int ap_id , unsigned char cr, unsigned char tafaddress) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Cr	unsigned char	I	Call number Specifies the number of a call to be forcibly disconnected.
Tafaddress	unsigned char	I	TAF address * Set 0.
Return value	Type	I/O	Description
Sts	Int	O	Result of processing ELIB_CP_OK: Successful ELIB_CP_NG: Unsuccessful
Include File	target/v4t_le/include/srv_cp.h		
Functional description	<p>This function makes a request for forcibly disconnecting PS. PS Service makes a notification of requesting PS forced disconnection to DCF. DCF-status (ForcedRelease) is transmitted.</p> <p>In the call number argument, specify the call number posted with a DCF/PS-TAF message.</p>		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example	None		

1.12 Get U-Plane connection status

1-12 Get U-Plane connection status			
Classification	PS Service/Packet communication watching		
Function	Get U-Plane connection status		
Symbol	Elib_CP_Get_UPlane_Status		
Syntax	int Elib_CP_Get_UPlane_Status (unsigned int ap_id);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Return value	Type	I/O	Description
Sts	Int	O	Packet communication status ELIB_CP_UPLANE_ON : U-Plane connected ELIB_CP_UPLANE_OFF: U-Plane not connected
Include File	target/v4t_le/include/srv_cp.h		
Functional description	This function gets the current U-Plane connection status.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example	None		

1.13 Request temporary change of APN settings

1-13 Request temporary change of APN settings			
Classification	PS Service/Packet communication watching		
Function	Request temporary change of APN settings		
Symbol	Elib_CP_CDF_APN_Set_Req		
Syntax	int Elib_CP_CDF_APN_Set_Req (unsigned int ap_id, , unsigned char * apn) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Apn	unsigned char *	O	APN data The allowable size is 1 to ELIB_CP_APN_MAX + NULL
Return value	Type	I/O	Description
Sts	Int	O	Result of processing ELIB_CP_OK: Successful ELIB_CP_NG: Unsuccessful
Include File	target/v4t_le/include/srv_cp.h		
Functional description	<p>This function makes a temporary change of APN settings while a certificate is downloaded.</p> <p>ELIB_CP_APN_MAX /* Maximum number of characters for user-specified APN data */</p> <p>Note) The service side does not check any user-specified APN data, thus the APL side needs to check the APN data code.</p> <p>Making a setting change by using this function requires "Request to resume temporary change of APN settings" to be executed to close.</p>		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example	None		

1.14 Request to resume temporary change of APN settings

1-14 Request to resume temporary change of APN settings			
Classification	PS Service/Packet communication watching		
Function	Request to resume temporary change of APN settings		
Symbol	Elib_CP_CDF_APN_Reset_Req		
Syntax	int Elib_CP_CDF_APN_Reset_Req (unsigned int ap_id);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Return value	Type	I/O	Description
Sts	Int	O	Result of processing ELIB_CP_OK: Successful ELIB_CP_NG: Unsuccessful
Include File	target/v4t_le/include/srv_cp.h		
Functional description	This function restores the APN settings to which a temporary change was applied while a certificate was downloaded.		
Related message	None		
Necessary procedure	None		
Note	"Request temporary change of APN settings" needs to be executed in advance.		
Prohibition	None		
Use example	None		

1.15 Get DNS record name

1-15 Get DNS record name			
Classification	PS Service/Packet communication watching		
Function	Get DNS record name		
Symbol	Elib_CP_Get_DNS_Record		
Syntax	int Elib_CP_Get_DNS_Record (unsigned int ap_id , unsigned char * DNS_Record) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
DNS_Record	unsigned char *	O	DNS record name The allowable size is ELIB_CP_DNS_RECORD_MAX + NULL.
Return value	Type	I/O	Description
Sts	Int	O	Result of processing ELIB_CP_OK: Successful ELIB_CP_NG: Unsuccessful
Include File	target/v4t_le/include/srv_cp.h		
Functional description	<p>This function gets the name of the DNS record for the destination which is being used.</p> <p>For a destination in a mobile station, it is fixed to "wpcg".</p> <p>When the default in an UIM and APN default settings (UIM data available and UIM card version 2) are used, it depends on the data in the UIM.</p>		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example	None		

1.16 Get default APN selection

1-16 Get default APN selection			
Classification	PS Service/Packet communication watching		
Function	Get default APN selection		
Symbol	Elib_CP_APN_Default_Mode		
Syntax	int Elib_CP_APN_Default_Mode (unsigned int ap_id);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Return value	Type	I/O	Description
Sts	Int	O	Selection ELIB_CP_DEFAULTAPN_MS: Settings in the mobile station ELIB_CP_DEFAULTAPN_UIM: Settings in the UIM
Include File	target/v4t_le/include/srv_cp.h		
Functional description	This function gets the type of default APN settings.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example	None		

1.17 Request to get WPCG IP address

1-17 Request to get WPCG IP address			
Classification	PS Service/Packet communication watching		
Function	Request to get WPCG IP address		
Symbol	Elib_CP_WPCG_IPAddress_Get		
Syntax	int Elib_CP_WPCG_IPAddress_Get(unsigned int ap_id , int *wpcg_data) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
wpcg_data	int	O	WPCG address storing pointer
Return value	Type	I/O	Description
Sts	int	O	Result of processing ELIB_CP_OK: Successful ELIB_CP_NG: Unsuccessful
Include File	target/v4t_le/include/srv_cp.h		
Functional description	This function returns the "wpcg" IP address for packet communication connection.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example	None		

1.18 Request to forcibly disconnect packet communication

1-18 Request to forcibly disconnect packet communication			
Classification	PS Service/Packet communication watching		
Function	Request to forcibly disconnect packet communication		
Symbol	Elib_CP_ShutDown_Req		
Syntax	int Elib_CP_ShutDown_Req(unsigned int ap_id) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Return value	Type	I/O	Description
Sts	int	O	Result of processing ELIB_CP_OK: Successful ELIB_CP_NG: Unsuccessful
Include File	target/v4t_le/include/srv_cp.h		
Functional description	<p>This function disconnects packet communication currently being used. For APL with no call number retained.</p> <p>Disconnect all PS calls (IPoPDCP) being used. For all PS calls (IPoPDCP), make a request to PS-TAF for AT-BIN(+CGACT=0).</p> <p>Difference from "A-11 Request to forcibly disconnect packet communication"</p> <p>Use a forced disconnection when communication of an external TE is disconnected by the mobile station power key.</p> <p>This function disconnects all active calls in a regular manner.</p>		
Related message	None		
Necessary procedure	None		
Note	<p>This function disconnects all active PS calls (IPoPDCP) regardless of circumstances, thus the calling side must solve contentions.</p> <p>* PPPoPDCP is controlled with AT-INT, thus this function does not handle any disconnection for it.</p>		
Prohibition	Any call from other than APL is prohibited.		
Use example	None		

1.19 Request to initialize APN

1-19 Request to initialize APN			
Classification	PS Service/External equipment interface connection status		
Function	Request to initialize APN		
Symbol	Elib_CP_APN_Init_Req		
Syntax	int Elib_CP_APN_Init_Req (unsigned int ap_id) ;		
Argument	Type	I/O	Description
Ap_id	unsigned int	I	Application ID
Return value	Type	I/O	Description
Sts	Int	O	Result of processing ELIB_CP_UIM_CHANGED : Normal...UIM exchanged (mismatched) ELIB_CP_UIM_NOCHANGED: Normal...UIM not exchanged (matched) ELIB_CP_NG : Error...Failure returned from the authentication service
Include File	target/v4t_le/include/srv_cp.h		
Functional description	<p>This function makes APN initialization.</p> <p>When the power is turned on, this function starts APN initialization according to a request from the authentication AP.</p> <p>If the UIM ID is not the same as that referenced last time, the default APN in the mobile station or default APN in the UIM is set according to some conditions. The destination to be connected is the default.</p> <p>When the UIM ID agrees with that referenced last time, the last UIM ID is preserved.</p> <p>In case an error such as no destination data can be read from nonvolatile memory or UIM, set up so that the default APN is to be used in the mobile station.</p> <p>The notification of APN initialization completed is posted as an event. Addition information to it is used to post the destination number used.</p>		
Related message	None		
Necessary procedure	To receive the APN initialization completed notification event, make a call to the "Start watching external equipment for packet communication" in advance in order to register a callback function.		
Note	None		
Prohibition	None		

Use example	None
-------------	------

1.20 Start watching external equipment for packet communication

1-20 Start watching external equipment for packet communication			
Classification	PS Service/External equipment interface connection status		
Function	Start watching external equipment for packet communication		
Symbol	Elib_CP_Start_ExtDevice_Watch		
Syntax	int Elib_CP_Start_ExtDevice_Watch (unsigned int ap_id , int mask, MsbFunc callback_func) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Mask	Int	I	Notification event mask Enabled with the bit turned ON ELIB_CP_MASK_APN_INIT_END 0x01: Notification of APN initialization completed ELIB_CP_MASK_ALL 0xff: All
callback_func	MsbFunc	I	Callback function of which the event is notified
Return value	Type	I/O	Description
Sts	Int	O	Result of processing ELIB_CP_OK: Successful ELIB_CP_NG: Unsuccessful
Include File	target/v4t_le/include/srv_cp.h		
Functional description	<p>This function starts giving notice about events related to external equipment for packet communication.</p> <p>This function notifies a requesting application of the following event:</p> <p><Notification event></p> <p><u>Notification of APN initialization completed:</u></p> <ul style="list-style-type: none"> - Gives notice of APN initialization completed and destination number to be used. <p><APN initialization completed notification event></p> <pre>typedef struct { int category ; int subtype ; int info ; int subinfo ; union { ... } data ; } _ELIB_CP_EVENT ;</pre> <p>Setting value type</p> <p>ELIB_CP_APN_INIT_END_IND: Notification of APN initialization completed</p> <p>Setting value</p>		

	ELIB_CP_APN_DEFAULT : Default setting ELIB_CP_APN_USERSET1 : User-specified 1 ELIB_CP_APN_USERSET2 : User-specified 2 ELIB_CP_APN_USERSET3 : User-specified 3 ELIB_CP_APN_USERSET4 : User-specified 4 ELIB_CP_APN_USERSET5 : User-specified 5 ELIB_CP_APN_USERSET6 : User-specified 6 ELIB_CP_APN_USERSET7 : User-specified 7 ELIB_CP_APN_USERSET8 : User-specified 8 ELIB_CP_APN_USERSET9 : User-specified 9 ELIB_CP_APN_USERSET10: User-specified 10 ELIB_CP_NG : Error
Related message	None
Necessary procedure	To receive the APN initialization completed notification event, make a call to "D-1 Request to initialize APN" after calling this function.
Note	None
Prohibition	None
Use example	None

1.21 Stop watching external equipment for packet communication

1-21 Stop watching external equipment for packet communication			
Classification	PS Service/External equipment interface connection status		
Function	Stop watching external equipment for packet communication		
Symbol	Elib_CP_Stop_ExtDevice_Watch		
Syntax	int Elib_CP_Stop_ExtDevice_Watch (unsigned int ap_id, int mask) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Mask	Int	I	Notification event mask Stops a message notification whose bit turned ON ELIB_CP_MASK_APN_INIT_END 0x01: Notification of APN initialization completed ELIB_CP_MASK_ALL 0xff: All
Return value	Type	I/O	Description
Sts	Int	O	Result of processing ELIB_CP_OK: Successful ELIB_CP_NG: Unsuccessful
Include File	target/v4t_le/include/srv_cp.h		
Functional description	This function stops giving notice of specified events. For details about notification events, see "D-4 Start watching external equipment for packet communication."		
Related message	None		
Necessary procedure	A request for notification events has been made by using the "Start watching external equipment for packet communication" in advance.		
Note	None		
Prohibition	None		
Use example	None		

2. Voice Communication Service Functions

2.1 Starting voice communication monitoring

2-1 Starting voice communication monitoring			
Classification	Voice communication service		
Function	Starting voice communication monitoring		
Symbol	Elib_CV_Start_Com_Watch		
Syntax	int Elib_CV_Start_Com_Watch (ap_id, mask, callbackFunc);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
mask	int	I	Mask of the notification event Starts an event notification with bit on. ELIB_CV_MASK_COM_STATUS 0x01 : Voice communication status notification ELIB_CV_MASK_TLK_TIME 0x02 : Call time notification ELIB_CV_MASK_DISC_CAUSE 0x04 : Disconnection cause notification ELIB_CV_MASK_FW_RESULT 0x08 : Forwarding result notification ELIB_CV_MASK_OFFHK_TO 0x20 : Off-hook transmission timeout notification ELIB_CV_MASK_SELF_MODE 0x80 : Self-mode icon notification ELIB_CV_MASK_ALL 0xff : All notified
callbackFunc	MsbFunc	I	Callback function the event is notified
Return value	Type	I/O	Description
Sts	int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal
Include file	srv_cv.h		
Functional description	<p>Starts notifying of an event about voice communication. Notifies the requesting application of the following events.</p> <p><Events to be reported></p> <p><u>Voice communication status notification:</u></p> <ul style="list-style-type: none"> - Notifies that the voice communication status is changed. <p><u>Call time notification:</u></p> <ul style="list-style-type: none"> - Notifies that the call time is changed. (unit: seconds) <p><u>Disconnection cause notification:</u></p> <ul style="list-style-type: none"> - Notifies of the cause why an outgoing call, an incoming call, and the current call are disconnected. <p><u>Forwarding result notification:</u></p> <ul style="list-style-type: none"> - Notifies of the results of forwarding an incoming call and forwarding a voice message. 		

Reject signal notification:

- Notifies that a reject signal is received during voice communication, USSD/SD/SS, or 64K communication/AV communication.

Off-hook transmission timeout notification:

- Notifies of timeout of the off-hook transmission timer (5 seconds). Timeout is notified when manual transmission is specified in A-6 Off-hook request.

Self-mode icon notification:

- Notifies of the event when self-mode setting is changed.

* To notify of an event, use MSB.

<Voice communication status notification>

Voice communication status notification event structure

```
typedef struct {
    int      category;  -> VoiceNotify
    int      subtype;   -> VoiceNotify_ConnInfo
    int      info;      -> Voice communication status
    int      subinfo;   -> Line type
    union {
        ...
    } data;             -> Unused
} _ELIB_CV_EVENT;
```

Voice communication status

ELIB_CV_COM_STATUS_WAIT	: Standby
ELIB_CV_COM_STATUS_RCV	: Under incoming
ELIB_CV_COM_STATUS_TRN	: Under outgoing
ELIB_CV_COM_STATUS_DLV	: Under calling
ELIB_CV_COM_STATUS_TLK	: Under conversation
ELIB_CV_COM_STATUS_HLD	: Under response hold
ELIB_CV_COM_STATUS_RLS	: Under release
ELIB_CV_COM_STATUS_TLK_RCV	: Under conversation and incoming
ELIB_CV_COM_STATUS_TLK_TRN	: Under conversation and outgoing
ELIB_CV_COM_STATUS_TLK_DLV	: Under conversation and calling
ELIB_CV_COM_STATUS_TLK_RSV	: Under conversation and hold
ELIB_CV_COM_STATUS_RSV_RLS	: Under hold and release
ELIB_CV_COM_STATUS_TLK_RSV_RCV	: Under conversation, hold, and incoming
ELIB_CV_COM_STATUS_RCV_AV	: Under incoming of an AV call
ELIB_CV_COM_STATUS_TRN_AV	: Under outgoing of an AV call
ELIB_CV_COM_STATUS_DLV_AV	: Under calling of an AV call
ELIB_CV_COM_STATUS_TLK_AV	: Under conversation of an AV call
ELIB_CV_COM_STATUS_HLD_AV	: Under response hold of an AV call
ELIB_CV_COM_STATUS_RLS_AV	: Under AV release
ELIB_CV_COM_STATUS_BSY	: Under busy on the other party (*)

<Call time notification>

Call time notification event structure

```
typedef struct {
    int      category;  -> VoiceNotify
    int      subtype;   -> VoiceNotify_TelCallTime
    int      info;       -> Call time (seconds)
    int      subinfo;    -> Unused

    union {
        ...
    } data;              -> Unused
} _ELIB_CV_EVENT;
```

<Disconnection cause notification>

Disconnection cause notification event structure

```
typedef struct {
    int      category;  -> VoiceNotify
    int      subtype;   -> VoiceNotify_DiscCause
    int      info;       -> Call number
    int      subinfo;    -> Unused

    union {
        _ELIB_CV_DISC_CAUSE cme; -> Disconnection cause information structure
    } data;
} _ELIB_CV_EVENT;
```

Disconnection cause information structure

```
typedef struct tagELIB_CV_DISC_CAUSE {
    unsigned char e_code; /* Result code flag */
    unsigned char code;   /* Result code */
    unsigned char e_cause1; /* Error cause 1 flag */
    unsigned char cause1;   /* Error cause 1 (ccpMtCause) */
    unsigned char e_cause2; /* Error cause 2 flag */
    unsigned char cause2;   /* Error cause 2 (Cause) */
} _ELIB_CV_DISC_CAUSE;
```

<Forwarding result notification>

Forwarding result notification event structure

```
typedef struct {
    int      category;  -> VoiceNotify
    int      subtype;   -> VoiceNotify_FW_Result
    int      info;       -> Call number
    int      subinfo;    -> Forwarding result

    union {
        _ELIB_CV_FW_RESULT fw_result; -> Forwarding result structure
    } data;
```

	<pre> } _ELIB_CV_EVENT ; Forwarding result ELIB_CV_OK Successful forwarding ELIB_CV_NG Forwarding failure Forwarding result structure typedef struct tagELIB_CV_FW_RESULT { int cause ; /* forwarding result details */ } _ELIB_CV_FW_RESULT ; Forwarding result details (* Set only at forwarding failure.) ELIB_CV_FW_ERROR_NO_JOIN Service is not contracted. ELIB_CV_FW_ERROR_NO_SETDATA The forwarded destination is not registered. ELIB_CV_FW_ERROR_ETC Others <Off-hook transmission timeout notification> Off-hook transmission timeout event structure typedef struct { int category; -> VoiceNotify int subtype; -> VoiceNotify_OffHk_Trn int info; -> Call number int subinfo; -> Communication type union { ... } data ; -> Unused } _ELIB_CV_EVENT ; Communication type ELIB_CV_BTYPE_CS_VOICE Voice ELIB_CV_BTYPE_CS_UD32 UD 32K communication ELIB_CV_BTYPE_CS_UD64 UD 64K communication ELIB_CV_BTYPE_CS_AV32 AV 32K communication ELIB_CV_BTYPE_CS_AV64 AV 64K communication </pre>
Related message	Message from CS-TAF and DCF
Necessary procedure	None
Note	None
Prohibition	None
Use example	

2.2 Ending voice Communication monitoring

2-2 Ending voice communication monitoring			
Classification	Voice communication service		
Function	Ending voice communication monitoring		
Symbol	Elib_CV_Stop_Com_Watch		
Syntax	int Elib_CV_Stop_Com_Watch (ap_id, mask);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Mask	int	I	Mask of the notification event to be deleted ELIB_CV_MASK_COM_STATUS 0x01 : Voice communication status notification ELIB_CV_MASK_TLK_TIME 0x02 : Call time notification ELIB_CV_MASK_DISC_CAUSE 0x04 : Disconnection cause notification ELIB_CV_MASK_FW_RESULT 0x08 : Forwarding result notification ELIB_CV_MASK_OFFHK_TO 0x20 : Off-hook transmission timeout notification ELIB_CV_MASK_SELF_MODE 0x80 : Self-mode icon notification ELIB_CV_MASK_ALL 0xff : All notified
Return value	Type	I/O	Description
Sts	int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal
Include file	srv_cv.h		
Functional description	Ends notifying of the event about specified voice communication. For notification events, see "A-1 Starting voice communication monitoring."		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.3 obtaining voice communication status

2-3 Obtaining the voice communication status			
Classification	Voice communication service		
Function	Obtaining the voice communication status		
Symbol	Elib_CV_Get_Com_Status		
Syntax	int Elib_CV_Get_Com_Status (ap_id);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Return value	Type	I/O	Description
Sts	Int	O	<p>Voice communication status</p> <p>ELIB_CV_COM_STATUS_WAIT : Standby</p> <p>ELIB_CV_COM_STATUS_RCV : Under incoming</p> <p>ELIB_CV_COM_STATUS_TRN : Under outgoing</p> <p>ELIB_CV_COM_STATUS_DLV : Under calling</p> <p>ELIB_CV_COM_STATUS_TLK : Under conversation</p> <p>ELIB_CV_COM_STATUS_HLD : Under response hold</p> <p>ELIB_CV_COM_STATUS_RLS : Under release</p> <p>ELIB_CV_COM_STATUS_TLK_RCV : Under conversation and incoming</p> <p>ELIB_CV_COM_STATUS_TLK_TRN : Under conversation and outgoing</p> <p>ELIB_CV_COM_STATUS_TLK_DLV : Under conversation and calling</p> <p>ELIB_CV_COM_STATUS_TLK_RSV : Under conversation and hold</p> <p>ELIB_CV_COM_STATUS_RSV_RLS : Under hold and release</p> <p>ELIB_CV_COM_STATUS_TLK_RSV_RCV : Under conversation, hold, and incoming</p> <p>ELIB_CV_COM_STATUS_RCV_AV : Under incoming of an AV call</p> <p>ELIB_CV_COM_STATUS_TRN_AV : Under outgoing of an AV call</p> <p>ELIB_CV_COM_STATUS_DLV_AV : Under calling of an AV call</p> <p>ELIB_CV_COM_STATUS_TLK_AV : Under conversation of an AV call</p> <p>ELIB_CV_COM_STATUS_HLD_AV : Under response hold of an AV call</p> <p>ELIB_CV_COM_STATUS_RLS_AV : Under AV release</p>
Include file	srv_cv.h		
Functional description	<p>Obtains the current voice communication status.</p> <p>Even if the voice communication monitoring start function is not called for monitoring, only the status can be obtained by this function.</p>		
Related message	None		
Necessar	None		

y proced ure	
Note	None
Prohibiti on	None
Use example	

2.4 Referring to connection destination information

2-4 Referring to connection destination information			
Classification	Voice communication service		
Function	Referring to connection destination information		
Symbol	Elib_CV_Connect_Info_Ref		
Syntax	int Elib_CV_Connect_Info_Ref (ap_id, call_No, connect_inf_p);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
call_No	int	I	Call number (0 to 255)
connect_inf_p	_ELIB_CV_CONNECT_INF *	O	Connection destination information pointer
<div> <div>CN_No</div> <div>CN_status</div> <div>status</div> <div>continue_flag</div> <div>Calling_Dail[ELIB_CV_DIAL_MAX+1]</div> <div>Called_Dail[ELIB_CV_DIAL_MAX+1]</div> </div>	int		Call number
	int		Call number status ELIB_CV_USED : Being used ELIB_CV_UNUSED : Unused
	int		Call status ELIB_CV_CHAN_KIND_NULL : Vacant ELIB_CV_CHAN_KIND_OFF : Off-hook ELIB_CV_CHAN_KIND_TRN : Outgoing call ELIB_CV_CHAN_KIND_DLV : Calling ELIB_CV_CHAN_KIND_RCV : Incoming call ELIB_CV_CHAN_KIND_REQ_T : Response (conversation) ELIB_CV_CHAN_KIND_ACT : Under conversation ELIB_CV_CHAN_KIND_REQ_H : Response (hold) ELIB_CV_CHAN_KIND_HLD : Response hold ELIB_CV_CHAN_KIND_RSV : Under hold ELIB_CV_CHAN_KIND_REL : Under release
	int		Existence of continuation data ELIB_CV_ON : Data with a number subsequent to the outgoing number exists. ELIB_CV_OFF : Data with a number subsequent to the outgoing number does not exist. Set to ELIB_CV_ON when the status is in the under incoming or under conversation and incoming status.
	unsigned char		Outgoing number dial "Other party number" is set. ELIB_CV_DIAL_MAX is 45.
	unsigned char		Incoming number dial

OwnNumber_Flg	unsigned char		Own number flag ELIB_CV_ON : Own number ELIB_CV_OFF : Not own number
BTsound_Inf	unsigned char		BT sound flag ELIB_CV_SOUND_BT_ON : BT tone sounds. ELIB_CV_SOUND_BT_OFF: BT tone is being stopped.
bc_type	int		Communication type ELIB_CV_BTTYPE_CS_NONE : None (unfixed) ELIB_CV_BTTYPE_CS_VOICE : Voice ELIB_CV_BTTYPE_CS_UD32 : UD 32K communication ELIB_CV_BTTYPE_CS_UD64 : UD 64K communication ELIB_CV_BTTYPE_CS_AV32 : AV 32K communication ELIB_CV_BTTYPE_CS_AV64 : AV 64K communication
tafaddress	unsigned char		TAF address (internal/external TAF type) 32 to 63 : Internal TAF 64 to 79 : External TAF
CauseofNoCLI	unsigned char		Blocking cause ELIB_CV_NOCL_NOSRV : No Notification can be given because the service cannot be supported. ELIB_CV_NOCL_USER : No notification can be given because the user rejection. ELIB_CV_NOCL_INTRACTSRV : No notification can be given because of service convergence. ELIB_CV_NOCL_PAYPHON : No notification can be given because the call is transmitted from a public phone.
num_presentatindicator	unsigned char		Display identifier ELIB_CV_PRSENT_IND_ALLOWED : Displayable ELIB_CV_PRSENT_IND_RESTRICTED : Display is impossible. ELIB_CV_PRSENT_IND_NOT_AVAILABLE : Displayable number does not exist. ELIB_CV_PRSENT_IND_RESERVE : Reservation
redirectnum[ELIB_CV_DIAL_M AX+1]	unsigned char		Redirect number (transfer source number)
redirect_presentatindicator	unsigned char		Redirect number display identifier ELIB_CV_PRSENT_IND_ALLOWED : Displayable ELIB_CV_PRSENT_IND_RESTRICTED : Display is impossible. ELIB_CV_PRSENT_IND_NOT_AVAILABLE : Displayable number does not exist. ELIB_CV_PRSENT_IND_RESERVE : Reservation
signal	unsigned char		Signal information ELIB_CV_SIGNAL_DIAL_TONE_ON : Dial tone on ELIB_CV_SIGNAL_RINGBACK_TONE_ON : Ring back tone on

			ELIB_CV_SIGNAL_INTERCEPT_TONE_ON : Intercept tone on ELIB_CV_SIGNAL_NW_CONGESTION_TONE_ON : Network congestion tone on ELIB_CV_SIGNAL_BUSY_TONE_ON : Busy tone on ELIB_CV_SIGNAL_CONFIRM_TONE_ON : Confirm tone on ELIB_CV_SIGNAL_ANSWER_TONE_ON : Answer tone on ELIB_CV_SIGNAL_CALLWAITING_TONE_ON : Call waiting tone on ELIB_CV_SIGNAL_OFFHK_WARNING_TONE_ON : Off-hook warning tone on ELIB_CV_SIGNAL_TONES_OFF : Tones off ELIB_CV_SIGNAL_ALERTING_OFF : Alerting off ELIB_CV_SIGNAL_UNSETTING : Signal information is not set.
cause	T_ELIB_CV_CME		CME structure
flag	unsigned int		Unused
sipurl[ELIB_CV_SIPURL_MAX]	unsigned char		Unused
dispname[ELIB_CV_SIPURL_MAX]	unsigned char		Unused
wlan_cause	unsigned char		Unused
Ringging_tone	unsigned char		Unused
Return value	Type	I/O	Description
sts	int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal
Include file	srv_cv.h		
Functional description	Refers to the connection destination information of the specified call number. The information can be referred to, even if the voice communication monitoring start function is not called for monitoring. - The connection destination information on up to three call numbers can be obtained. (Since up to three secured connection destinations are returned as information on the call numbers of the argument, the connection destination information on only a call can be obtained at a time.) - In the following cases, NG is returned. 1. The same call number does not exist. 2. Others parameter NG		
Related message	None		
Necessary procedure	None		

Note	
Prohibition	None
Use example	

2.5 Obtaining the length of the current call

2-5 Obtaining the length of the current call			
Classification	Voice communication service		
Function	Obtaining the length of the current call		
Symbol	Elib_CV_Get_Now_Talk_Time		
Syntax	unsigned int Elib_CV_Get_Now_Talk_Time(ap_id);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Return value	Type	I/O	Description
talk_time	unsigned int	O	Current call time (unit: seconds)
Include file	srv_cv.h		
Functional description	Obtains the call time counted by the voice communication service. When no call exists, 0 is returned. Operation when call time exceeds the maximum value of unsigned int is not secured currently.		
Related message	None		
Necessary procedure			
Note	None		
Prohibition	None		
Use example			

2.6 Off-hook request

2-6 Off-hook request			
Classification	Voice communication service		
Function	Off-hook request		
Symbol	Elib_CV_OffHook_Req		
Syntax	void Elib_CV_OffHook_Req (ap_id, type, option) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Type	int	I	Communication type ELIB_CV_BTTYPE_CS_VOICE Voice ELIB_CV_BTTYPE_CS_UD32 UD 32K communication ELIB_CV_BTTYPE_CS_UD64 UD 64K communication ELIB_CV_BTTYPE_CS_AV32 AV 32K communication ELIB_CV_BTTYPE_CS_AV64 AV 64K communication
Option	int	I	Option ELIB_CV_OFFHK_AUTO Automatic transmission ELIB_CV_OFFHK_MANUAL Manual transmission
Return value	Type		Description
None	Void		
Include file	srv_cv.h		
Functional description	Requests off-hook operation. The function is an immediate return function, and sends a message to a service and returns. To confirm the results, issue "Starting voice communication monitoring" to obtain the communication status. Standby : DT sounds. Others : Invalid		
	Operation at timer timeout (5 seconds) varies depending on the specification of the option.		
	Option	Operation at timer timeout (5 seconds)	
	ELIB_CV_OFFHK_AUTO (automatic transmission)	Automatic transmission operation is performed by the dials input in "Dial "	
	ELIB_CV_OFFHK_MANUAL (manual transmission)	Notifies of timeout to an application, and waits for the notification of transmission start from the application ("Dial completion" or " Requesting on-hook transmission"). Timeout is notified by monitoring " off-hook transmission timeout notification" in " Starting voice communication monitoring."	
	Transmission operation is started by calling either of the following functions after off-hook. For "Dial completion" - Transmits a call by the dials input in " Dial."		
	For "Requesting on-hook transmission"		

	<ul style="list-style-type: none"> - The dials input in "Dial" are discarded, and the dials given by the API are used for transmission. <p>When a mobile device is moved to low voltage, a low voltage notification is sent from DCF.</p>
Related message	None
Necessary procedure	To confirm the results, issue "Starting voice communication monitoring" to obtain the communication status.
Note	<ul style="list-style-type: none"> - During low voltage, when the communication status is other than the under hold status, an off-hook request is disabled. - If a call comes during off-hook, off-hook is canceled. - If you specify a subaddress, specify manual transmission for the option and use " Requesting on-hook transmission."
Prohibition	None
Use example	

2.7 Disconnect request

2-7 Disconnection request			
Classification	Voice communication service		
Function	Disconnection request		
Symbol	Elib_CV_Disc_Req		
Syntax	void Elib_CV_Disc_Req (ap_id, type, dummy) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Type	int	I	<p>Communication type</p> <p>ELIB_CV_BTYPE_ANY No specification (The service side determines depending on the communication status.)</p> <p>ELIB_CV_BTYPE_CS_VOICE Voice</p> <p>ELIB_CV_BTYPE_CS_UD32 UD 32K communication</p> <p>ELIB_CV_BTYPE_CS_UD64 UD 64K communication</p> <p>ELIB_CV_BTYPE_CS_AV32 AV 32K communication</p> <p>ELIB_CV_BTYPE_CS_AV64 AV 64K communication</p>
Dummy	int	I	Specify 0. (fixed)
Return value	Type	I/O	Description
None	void	-	
Include file	srv_cv.h		
Functional description	<p>Requests to disconnection the call.</p> <p>This function is an immediate return function, and sends a message to a service and returns.</p>		
Related message	None		
Necessary procedure	To confirm the results, issue "A-1 Starting voice communication monitoring" to obtain the communication status.		
Note	<ul style="list-style-type: none"> - An incoming call cannot be disconnected. (Use "A-14 Rejecting an incoming call.") 		

	- If multiple calls exist, all calls are disconnected.
Prohibition	None
Use example	

2.8 Dial

2-8 Dial			
Classification	Voice communication service		
Function	Dial		
Symbol	Elib_CV_Dial		
Syntax	int Elib_CV_Dial(ap_id, type, dial_buf, dial_len);		
Argument	Type	I/O	Description
Ap_id	unsigned int	I	Application ID
Type	int	I	Communication type ELIB_CV_BTYPE_CS_VOICE Voice ELIB_CV_BTYPE_CS_UD32 UD 32K communication ELIB_CV_BTYPE_CS_UD64 UD 64K communication ELIB_CV_BTYPE_CS_AV32 AV 32K communication ELIB_CV_BTYPE_CS_AV64 AV 64K COMMUNICATION
Dial_buf	unsigned char*	I	Dial data buffer address
Dial_len	int	I	Dial data length
Return value	Type	I/O	Description
Sts	int		Processing result ELIB_CV_OK : Normal
Include file	srv_cv.h		
Functional description	<p>Requests dialing.</p> <p>This function is an immediate return function, and sends a message to a service and returns.</p> <p>To confirm the results, issue "A-1 Starting voice communication monitoring" to obtain the communication status.</p> <p>The dial data stores the following ASCII codes.</p> <pre> 0: 0 x 30 1: 0 x 31 2: 0 x 32 3: 0 x 33 4: 0 x 34 5: 0 x 35 6: 0 x 36 7: 0 x 37 8: 0 x 38 9: 0 x 39 *: 0 x 2a #: 0 x 23 </pre> <p>Under off-hook</p> <p>Starts transmission operation with Dial + Dial completion.</p> <p>Five seconds later from the last dialing, operation is moved to the transmission</p>		

	status automatically. (When automatic transmission is specified in "A-6 Off-hook request.") Under on-hook DTMF is sent if the status is in the under conversation or under conversation and hold status.
Related message	None
Necessary procedure	To confirm the results, issue "A-1 Starting voice communication monitoring" to obtain the communication status. (The communication status changes by automatic transmission due to 5-second timeout, disconnection, and the like until completion of dialing.)
Note	
Prohibition	None
Use example	

2.9 Dial completion

2-9 Dial completion			
Classification	Voice communication service		
Function	Dial completion		
Symbol	Elib_CV_Dial_End		
Syntax	void Elib_CV_Dial_End(ap_id, type) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Type	int	I	Communication type ELIB_CV_BTTYPE_CS_VOICE Voice ELIB_CV_BTTYPE_CS_UD32 UD 32K communication ELIB_CV_BTTYPE_CS_UD64 UD 64K communication ELIB_CV_BTTYPE_CS_AV32 AV 32K communication ELIB_CV_BTTYPE_CS_AV64 AV 64K communication
Return value	Type	I/O	Description
None	void	-	
Include file	srv_cv.h		
Functional description	<p>Requests the end of inputting dials.</p> <p>This function is an immediate return function, and sends a message to a service and returns.</p> <p>To confirm the results, issue "A-1 Starting voice communication monitoring" to obtain the communication status.</p> <p>Under off-hook Starts transmission operation with Dial (A-8) + Dial completion.</p> <p>Under on-hook Disabled</p>		
Related message	None		

Necessary procedure	After calling "A-9 Dial function," transmission operation is performed by calling this function.
Note	None
Prohibition	None
Use example	

2.10 Response to an incoing call

2-10 Response to an incoming call			
Classification	Voice communication service		
Function	Response to an incoming call		
Symbol	Elib_CV_Rcv_Rsp		
Syntax	void Elib_CV_Rcv_Rsp (ap_id, type, dummy);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Type	Int	I	Communication type ELIB_CV_BTTYPE_ANY No specification (The service side determines depending on the communication status.) ELIB_CV_BTTYPE_CS_VOICE Voice ELIB_CV_BTTYPE_CS_UD32 UD 32K communication ELIB_CV_BTTYPE_CS_UD64 UD 64K communication ELIB_CV_BTTYPE_CS_AV32 AV 32K communication ELIB_CV_BTTYPE_CS_AV64 AV 64K communication
Dummy	Int	I	Specify 0. (fixed)
Return value	Type	I/O	Description
None	Void	-	
Include file	srv_cv.h		
Functional description	<p>Requests a response to an incoming call.</p> <p>This function is an immediate return function, and sends a message to a service and returns.</p> <p>To confirm the results, issue "A-1 Starting voice communication monitoring" to obtain the communication status.</p> <p>One of the following operations is performed depending on the status.</p> <p>Under incoming : Responds to an incoming call.</p> <p>Response hold : Responds to a response hold call (releasing response hold).</p> <p>Others : Disabled</p>		
Related message	None		
Necessary procedure	To confirm the results, issue "Starting voice communication monitoring" to obtain the communication status.		
Note	Disabled during low voltage.		
Prohibition	None		
Use example			

2.11 Forwarding an incoming call

2-11 Forwarding an incoming call			
Classification	Voice communication service		
Function	Forwarding an incoming call		
Symbol	Elib_CV_Rcv_Forward		
Syntax	void Elib_CV_Rcv_Forward(type);		
Argument	Type	I/O	Description
type	int	I	<p>Communication type</p> <p>ELIB_CV_BTTYPE_ANY No specification (The service side determines depending on the communication status.)</p> <p>ELIB_CV_BTTYPE_CS_VOICE Voice</p> <p>ELIB_CV_BTTYPE_CS_UD32 UD 32K communication</p> <p>ELIB_CV_BTTYPE_CS_UD64 UD 64K communication</p> <p>ELIB_CV_BTTYPE_CS_AV32 AV 32K communication</p> <p>ELIB_CV_BTTYPE_CS_AV64 AV 64K communication</p>
Return value	Type	I/O	Description
None	void	-	
Include file	srv_cv.h		
Functional description	<p>Requests to forward an incoming call.</p> <p>This function is an immediate return function, and sends a message to a network service and returns.</p> <p>To confirm the results, issue "A-1 Starting voice communication monitoring" to obtain the communication status.</p> <p>Forwards an incoming call, if the status is in the under incoming, under conversation and incoming, or under hold and incoming (incoming while a conversation call is disconnected from a three-party call) status.</p> <p>If forwarding fails, incoming is continued.</p>		
Related message	None		
Necessary procedure	To confirm the results, issue "A-1 Starting voice communication monitorin" to obtain the communication status.		
Note	None		
Prohibition	None		
Use example			

2.12 Phone-answering machine

2-12 Phone-answering machine			
Classification	Voice communication service		
Function	Phone-answering machine		
Symbol	Elib_CV_Voice_Msg		
Syntax	void Elib_CV_Voice_Msg(type);		
Argument	Type	I/O	Description
Type	int	I	<p>Communication type</p> <p>ELIB_CV_BTYPE_ANY No specification (The service side determines depending on the communication status.)</p> <p>ELIB_CV_BTYPE_CS_VOICE Voice</p> <p>ELIB_CV_BTYPE_CS_UD32 UD 32K communication</p> <p>ELIB_CV_BTYPE_CS_UD64 UD 64K communication</p> <p>ELIB_CV_BTYPE_CS_AV32 AV 32K communication</p> <p>ELIB_CV_BTYPE_CS_AV64 AV 64K communication</p>
Return value	Type	I/O	Description
None	void		
Include file	srv_cv.h		
Functional description	<p>Requests to forward a call to a phone-answering machine.</p> <p>This function is an immediate return function, and sends a message to a network and returns.</p> <p>To confirm the results, issue "A-1 Starting voice communication monitoring" to obtain the communication status.</p> <p>Forwards an incoming call to a phone-answering machine, if the status is in the under incoming, under conversation and incoming, or under hold and incoming (incoming while a conversation call is disconnected from a three-party call) status.</p> <p>If forwarding fails, incoming is continued.</p>		
Related message	None		
Necessary procedure	To confirm the results, issue "Starting voice communication monitoring" to obtain the communication status.		
Note	None		
Prohibition	None		
Use example			

2.13 Response hold

2-13 Response hold			
Classification	Voice communication service		
Function	Response hold		
Symbol	Elib_CV_Hold		
Syntax	void Elib_CV_Hold(type);		
Argument	Type	I/O	Description
type	int	I	<p>Communication type</p> <p>ELIB_CV_BTYPE_ANY No specification (The service side determines depending on the communication status.)</p> <p>ELIB_CV_BTYPE_CS_VOICE Voice</p> <p>ELIB_CV_BTYPE_CS_UD32 UD 32K communication</p> <p>ELIB_CV_BTYPE_CS_UD64 UD 64K communication</p> <p>ELIB_CV_BTYPE_CS_AV32 AV 32K communication</p> <p>ELIB_CV_BTYPE_CS_AV64 AV 64K communication</p>
Return value	Type	I/O	Description
None	void	-	-
Include file	srv_cv.h		
Functional description	<p>Requests response hold.</p> <p>To confirm the results, issue "A-1 Starting voice communication monitoring" to obtain the communication status.</p> <p>Either of the following operations is performed depending on the status.</p> <p>Under incoming : Response hold is performed for an incoming call.</p> <p>Others : Disabled</p>		
Related message	None		
Necessary procedure	To confirm the results, issue "Starting voice communication monitoring" to obtain the communication status.		
Note	To release response hold (move to the under conversation status) after moving to response hold, call "Response to an incoming call."		
Prohibition	None		
Use example			

2.14 Rejection an incoming call

2-14 Rejecting an incoming call			
Classification	Voice communication service		
Function	Rejecting an incoming call		
Symbol	Elib_CV_Rcv_Denial		
Syntax	void Elib_CV_Rcv_Denial(type);		
Argument	Type	I/O	Description
Type	int	I	<p>Communication type</p> <p>ELIB_CV_BTTYPE_ANY No specification (The service side determines depending on the communication status.)</p> <p>ELIB_CV_BTTYPE_CS_VOICE Voice</p> <p>ELIB_CV_BTTYPE_CS_UD32 UD 32K communication</p> <p>ELIB_CV_BTTYPE_CS_UD64 UD 64K communication</p> <p>ELIB_CV_BTTYPE_CS_AV32 AV 32K communication</p> <p>ELIB_CV_BTTYPE_CS_AV64 AV 64K communication</p>
Return value	Type	I/O	Description
None	void	-	
Include file	srv_cv.h		
Functional description	<p>Requests to reject an incoming call.</p> <p>This function is an immediate return function, and sends a message to a service and returns.</p> <p>To confirm the results, issue " Starting voice communication monitoring" to obtain the communication status.</p> <p>Operation for each status is as follows:</p> <p>Under incoming : Rejects an incoming call</p> <p>Under conversation and incoming : Rejects an incoming call</p> <p>Under hold and incoming : Rejects an incoming call</p> <p>Under conversation, hold, and incoming : Rejects an incoming call</p>		
Related message	None		
Necessary procedure	To confirm the results, issue "A-1 Starting voice communication monitoring" to obtain the communication status.		
Note	None		
Prohibition	None		
Use example			

2.15 Processing of operating multiple call

2-15 Processing of operating multiple calls			
Classification	Voice communication service		
Function	Processing of operating multiple calls		
Symbol	Elib_CV_CW_Op		
Syntax	void Elib_CV_CW_Op (type, kind, cr);		
Argument	Type	I/O	Description
Type	int	I	<p>Communication type</p> <p>ELIB_CV_BTTYPE_ANY No specification (The service side determines depending on the communication status.)</p> <p>ELIB_CV_BTTYPE_CS_VOICE Voice</p> <p>ELIB_CV_BTTYPE_CS_UD32 UD 32K communication</p> <p>ELIB_CV_BTTYPE_CS_UD64 UD 64K communication</p> <p>ELIB_CV_BTTYPE_CS_AV32 AV 32K communication</p> <p>ELIB_CV_BTTYPE_CS_AV64 AV 64K communication</p>
Kind	int	I	<p>Operation type</p> <p>ELIB_CV_MOP_RSV_DISC Disconnection of a hold call</p> <p>ELIB_CV_MOP_DISC_AND_RSP Response after disconnection</p> <p>ELIB_CV_MOP_RSV_AND_RSP Response after hold (including operation for switching a call)</p> <p>ELIB_CV_MOP_CR_DISC Disconnection by specifying a call number</p>
Cr	int	I	<p>Call number of the call to be disconnected</p> <p>Enabled only when ELIB_CV_MOP_CR_DISC is specified for the second argument. In other cases, the specification is ignored.</p>
Return value	Type	I/O	Description
None	void	-	-
Include file	srv_cv.h		
Functional description	<p>Requests operation for each call, when communication is made with multiple calls. Operation for each operation type is as follows:</p> <ul style="list-style-type: none"> - ELIB_CV_MOP_RSV_DISC Disconnects a hold call. - ELIB_CV_MOP_DISC_AND_RSP Disconnects a call and responds to another call (a incoming call or hold call). <ul style="list-style-type: none"> * Responds to an incoming call after disconnecting a call while the third call is incoming (a call, hold call, or incoming call). * Operation is not performed while only a hold call is incoming. - ELIB_CV_MOP_RSV_AND_RSP Holds a conversation call, and responds to another call (an incoming call or hold call). <ul style="list-style-type: none"> * Switches between a call and a hold call while the third call (a call, hold call, or 		

	<p>incoming call) is incoming.</p> <ul style="list-style-type: none"> * Responds to a hold call while only a hold call is incoming. <p>- ELIB_CV_MOP_CR_DISC</p> <p>Specify the call number to disconnect the call.</p> <p>To confirm the results, issue "Starting voice communication monitoring" to obtain the communication status.</p>
Related message	None
Necessary procedure	To confirm the results, issue "A-1 Starting voice communication monitoring" to obtain the communication status.
Note	None
Prohibition	None
Use example	

2.16 Requesting on-hook transmission

2-16 Requesting on-hook transmission			
Classification	Voice communication service		
Function	Requesting on-hook transmission		
Symbol	Elib_CV_Connect_Req		
Syntax	int Elib_CV_Connect_Req (ap_id, connect_req_p);		
Argument	Type	I/O	Description
Ap_id	unsigned int	I	Application ID
connect_req_p	_ELIB_CV_CONNECT_REQ *	I	Pointer to the transmission request structure
	Type	int	<p>Communication type</p> <p>ELIB_CV_BTYPE_CS_VOICE Voice</p> <p>ELIB_CV_BTYPE_CS_UD32 UD 32K communication</p> <p>ELIB_CV_BTYPE_CS_UD64 UD 64K communication</p> <p>ELIB_CV_BTYPE_CS_AV32 AV 32K communication</p> <p>ELIB_CV_BTYPE_CS_AV64 AV 64K communication</p>
	Dial_buf	unsigned char *	Dial data buffer address
	Dial_len	int	Dial data length
	Notice	int	<p>Sets whether the outgoing number is notified or blocked for each call.</p> <p>ELIB_CV_NOTICE_ON : Notified</p> <p>ELIB_CV_NOTICE_OFF : Blocked</p> <p>ELIB_CV_NOTICE_NOSET : No setting</p>
	subaddr_buf	unsigned char *	Subaddress data buffer address
	subaddr_len	i	Subaddress data length
Return value	Type	I/O	Description
Sts	i	O	<p>Processing result</p> <p>ELIB_CV_OK : Normal</p> <p>ELIB_CV_ONHOOK_DENY :</p>

			On-hook is possible. ELIB_CV_ONHOOK_STATUS_NG : NG due to communication convergence ELIB_CV_ONHOOK_OB_CR : Excess of the maximum number of call numbers ELIB_CV_NG : Abnormal (e.g. parameter error)
Include file	srv_cv.h		
Functional description	Requests transmission to the specified dial number (on-hook transmission) when the communication status is standby. Dial data stores ASCII codes. 0 : 0 x 30 1 : 0 x 31 2 : 0 x 32 3 : 0 x 33 4 : 0 x 34 5 : 0 x 35 6 : 0 x 36 7 : 0 x 37 8 : 0 x 38 9 : 0 x 39 * : 0 x 2a # : 0 x 23		
Related message	None		
Necessary procedure	None		
Note	- A transmission request during low voltage is disabled. - If "184" or "186" is placed at the head of dial data, the character string is deleted.		
Prohibition	None		
Use example			

2.17 Processing of obtaining the call number

2-17 Processing of obtaining the call number				
Classification		Voice communication service		
Function		Processing of obtaining the call number		
Symbol		Elib_CV_Get_ChanelNum		
Syntax		int Elib_CV_Get_ChanelNum(ap_id, ChanNum);		
Argument		Type	I/O	Description
ap_id		unsigned int	I	Application ID
ChanNum		_ELIB_CV_CHANNUM_I NF *	O	The head address of the call number information structure
	ChanNum_00	int		Call number information 00
	ChanNum_01	int		Call number information 01
	ChanNum_02	int		Call number information 02
Return value		Type	I/O	Description

Sts	int	O	Processing result ELIB_CV_OK: Normal end ELIB_CV_NG: Abnormal end
Include file	srv_cv.h		
Functional description	Obtains the call number in use. - A value within 0 to 255 is set to the call number. - If channel is not used, ELIB_CV_CHAN_NOUSE is set to the call number.		
Related message	None		
Necessary procedure	None		
Note			
Prohibition	None		
Use example			

2.18 Starting DCF message monitoring

2-18 Starting DCF message monitoring			
Classification	Voice communication service		
Function	Starting DCF message monitoring		
Symbol	Elib_CV_Start_DCF_Msg_Watch		
Syntax	int Elib_CV_Start_DCF_Msg_Watch (ap_id, mask, callbackFunc);		
Argument	Type	I/O	Description
Ap_id	unsigned int	I	Application ID
mask	int	I	<p>Mask of the communication type Starts an event notification with bit on.</p> <p>ELIB_CV_MASK_DCF_DISP 0x01 Display-related message ELIB_CV_MASK_DCF_HISTORY 0x02 History-related message ELIB_CV_MASK_DCF_TONE1 0x04 Tone 1-related message ELIB_CV_MASK_DCF_TONE2 0x08 Tone 2-related message ELIB_CV_MASK_DCF_ETC 0x80 Other messages ELIB_CV_MASK_ALL 0xff: All notified</p>
callbackFunc	MsbFunc	I	Callback function the event is notified
Return value	Type	I/O	Description
sts		O	<p>Processing result</p> <p>ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal</p>
Include file	srv_cv.h		
Functional description	<p>Starts notifying of a DCF message.</p> <p>Starts a notification of voice communication or AV communication (internal). * For messages of UDI communication and AV communication (external), see Starting external communication related message monitoring.</p> <p>Events to be notified</p> <p><u>Display-related message:</u></p> <ul style="list-style-type: none"> - Notification of starting display during CCP outgoing <ul style="list-style-type: none"> - Notifies of the start of display during CCP outgoing. - Notification of starting display during CCP incoming <ul style="list-style-type: none"> - Notifies of the start of display during CCP incoming. - Notification of starting display during CCP calling <ul style="list-style-type: none"> - Notifies of the start of display during CCP calling. - Notification of starting display during CCP connection <ul style="list-style-type: none"> - Notifies of the start of display during CCP connection. - Notification of starting display during CCP communication <ul style="list-style-type: none"> - Notifies of the start of display during CCP communication. 		

	<pre> int subinfo; Bearer type union { ... <DCF message structure corresponding to report types (*1)> ; ->DCF messages ... } data ; } ELIB_CV_EVENT ; </pre>	
	<pre> Event type VoiceNotify_DCF_Dis Display-related message VoiceNotify_DCF_History History-related message VoiceNotify_DCF_Tone1 Tone 1-related message VoiceNotify_DCF_Tone2 Tone 2-related message VoiceNotify_DCF_ETC Other messages </pre>	
	<pre> Notification type ELIB_CV_CCP_CALLING_START_REQ Notification of starting display during CCP outgoing ELIB_CV_CCP_CALLED_START_IND Notification of starting display during CCP incoming ELIB_CV_CCP_CALLING_ALERTING_IND Notification of starting display during CCP calling ELIB_CV_CCP_CONNECT_START_RSP Notification of starting display during CCP connection ELIB_CV_CCP_CONNECT_START_IND Notification of starting display during CCP communication ELIB_CV_CCP_RELEASE_IND Notification of ending CCP display ELIB_CV_CCP_DISCONNECT_REQ Notification of starting CCP disconnection (on a mobile device) display ELIB_CV_CCP_DISCONNECT_START_IND Notification of starting CCP disconnection (on a network) display ELIB_CV_CCP_CALLING_REJ_IND Notification of rejecting CCP outgoing ELIB_CV_CCP_HOLD_CNF Notification of CCP hold ELIB_CV_CCP_RETREIVE_CNF Notification of releasing CCP hold </pre>	
	<pre> ELIB_CV_CCP_CALLING_SETUP_REQ Notification of registering CCP outgoing call history ELIB_CV_CCP_CALLED_REJ_REQ Notification of registering CCP absence incoming call history ELIB_CV_CCP_CALLED_SETUP_RSP Notification of registering CCP incoming call history ELIB_CV_CCP_RGT_START Notification of CCP RGT start ELIB_CV_CCP_RGT_STOP Notification of CCP RGT stop ELIB_CV_CCP_HRGT_START Start notification of incoming of a CCP hold call ELIB_CV_CCP_HRGT_STOP Stop notification of incoming of a CCP hold call </pre>	
	<pre> ELIB_CV_CCP_DST_START Notification of CCP DST start ELIB_CV_CCP_DST_STOP Notification of CCP DST stop ELIB_CV_CCP_RBT_START Notification of CCP RBT start ELIB_CV_CCP_RBT_STOP Notification of CCP RBT stop ELIB_CV_CCP_BT_START Notification of CCP BT start ELIB_CV_CCP_CWT_START Notification of CCP CWT start ELIB_CV_CCP_CWT_STOP Notification of CCP CWT stop </pre>	
	<pre> ELIB_CV_CCP_REJECT_ASK Inquiry report of rejecting a CCP CS incoming call </pre>	
	<pre> Bearer type ELIB_CV_BTTYPE_CS_NONE None (unfixed) </pre>	

```

ELIB_CV_BTTYPE_CS_VOICE  AMR voice
ELIB_CV_BTTYPE_CS_UD32   Unlimited digital 32K communication
ELIB_CV_BTTYPE_CS_UD64   Unlimited digital 64K communication
ELIB_CV_BTTYPE_CS_AV32   AV 32K communication
ELIB_CV_BTTYPE_CS_AV64   AV 64K communication

*1) DCF message structure
union {

    ...

    T_ELIB_CV_CCP_CALLING_START_REQ  ccp_calling_start_req ; /* Notification of
starting display during CCP outgoing */
    T_ELIB_CV_CCP_CALLED_START_IND   ccp_called_start_ind ; /* Notification of
starting display during CCP incoming */
    T_ELIB_CV_CCP_CALLING_ALERTING_IND ccp_calling_alerting_ind /* Notification
of starting display during CCP calling */
    T_ELIB_CV_CCP_CONNECT_START_RSP  ccp_connect_start_rsp ; /* Notification of
starting display during CCP connection */
    T_ELIB_CV_CCP_CONNECT_START_IND  ccp_connect_start_ind ; /* Notification
of starting display during CCP communication */
    T_ELIB_CV_CCP_RELEASE_IND        ccp_release_ind ; /*
Notification of ending CCP display */
    T_ELIB_CV_CCP_DISCONNECT_REQ     ccp_disconnect_req ; /* Notification of
starting display of CCP disconnection (on a mobile device) display */
    T_ELIB_CV_CCP_DISCONNECT_START_IND ccp_disconnect_start_ind ; /*
Notification of starting display of CCP disconnection (on a network) display */
    T_ELIB_CV_CCP_CALLING_REJ_IND    ccp_calling_rej_ind ; /* Notification of
rejecting CCP outgoing */
    T_ELIB_CV_CCP_HOLD_CNF           ccp_hold_cnf ; /*
Notification of CCP hold */
    T_ELIB_CV_CCP_RETREIVE_CNF       ccp_retreive_cnf ; /*
Notification of releasing CCP hold */

    T_ELIB_CV_CCP_CALLING_SETUP_REQ  ccp_calling_setup_req ; /* Notification of
registering CCP outgoing call history */
    T_ELIB_CV_CCP_CALLED_REJ_REQ     ccp_called_rej_req ; /* Notification of
registering CCP absence incoming call history */
    T_ELIB_CV_CCP_CALLED_SETUP_RSP   ccp_called_setup_rsp ; /* Notification of
registering CCP incoming call history */

    T_ELIB_CV_CCP_RGT_START          ccp_rgt_start ; /*
Notification of CCP RGT start */
    T_ELIB_CV_CCP_RGT_STOP           ccp_rgt_stop ; /*
Notification of CCP RGT stop */
    T_ELIB_CV_CCP_HRGT_START         ccp_hrgt_start ; /* Start notification of
incoming of a CCP hold call */
    T_ELIB_CV_CCP_HRGT_STOP         ccp_hrgt_stop ; /* Stop notification of
incoming of a CCP hold call */

    T_ELIB_CV_CCP_DST_START          ccp_dst_start ; /* Notification of
CCP DST start */
    T_ELIB_CV_CCP_DST_STOP           ccp_dst_stop ; /* Notification of
CCP DST stop */
    T_ELIB_CV_CCP_RBT_START          ccp_rbt_start ; /* Notification of
CCP RBT start */
    T_ELIB_CV_CCP_RBT_STOP           ccp_rbt_stop ; /* Notification of
CCP RBT stop */
    T_ELIB_CV_CCP_BT_START           ccp_bt_start ; /* Notification
of CCP BT start */
    T_ELIB_CV_CCP_CWT_START          ccp_cwt_start ; /* Notification of
CCP CWT start */
    T_ELIB_CV_CCP_CWT_STOP           ccp_cwt_stop ; /* Notification of
CCP CWT stop */

```

	<pre>T_ELIB_CV_CCP_REJECT_ASK ccp_reject_sak ; /* Inquiry report of rejecting a CCP CS incoming call */ ... } data ;</pre>
Related message	Messages from DCF (voice communication or AV communication (internal))
Necessary procedure	None
Note	None
Prohibition	None
Use example	

2.19 Ending DCF message monitoring

2-19 Ending DCF message monitoring			
Classification	Voice communication service		
Function	DCF Message		
Symbol	Elib_CV_Stop_DCF_Msg_Watch		
Syntax	int Elib_CV_Stop_DCF_Msg_Watch (ap_id, mask);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
mask	Int	I	Mask of the communication type Ends an event notification with bit on. ELIB_CV_MASK_DCF_DISP 0x01 Display-related message ELIB_CV_MASK_DCF_HISTORY 0x02 History-related message ELIB_CV_MASK_DCF_TONE1 0x04 Tone 1-related message ELIB_CV_MASK_DCF_TONE2 0x08 Tone 2-related message ELIB_CV_MASK_DCF_ETC 0x80 Other messages ELIB_CV_MASK_ALL 0xff: All notified
Return value	Type	I/O	Description
Sts	Int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal
Include file	Srv_cv.h		
Functional description	Ends the notification of a DCF message (voice communication or AV communication (internal)). * For messages of UDI communication and AV communication (external), see Ending external communication related message monitoring.		
Related message	Messages from DCF (voice communication or AV communication (internal))		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.20 Processing of notifying start/stop a voice message

2-20 Processing of notifying of start/stop of a voice message			
Classification	Voice communication service		
Function	Processing of notifying of start/stop of a voice message		
Symbol	Elib_CV_Rec_Memo_Notify		
Syntax	Void Elib_CV_Rec_Memo_Notify (rec_status);		
Argument	Type	I/O	Description
rec_status	int	I	ELIB_CV_REC_MESSAGE_START: Start of a voice message ELIB_CV_REC_MESSAGE_STOP : Stop of a voice message
Return value	Type	I/O	Description
-	void	-	
Include file	SRV_CV.H		
Functional description	<p>Performs calling when a hold tone does not sound at the play of a voice response message.</p> <p>Be sure to perform calling before the communication status is moved to "under conversation."</p> <p>After the start notification, be sure to issue the stop notification when a voice message is stopped.</p>		
Related message	None		
Necessary procedure	None		
Note	- Only the telephone AP should use this function.		
Prohibition	None		
Use example			

2.21 Start processing of sounding a hold tone during a call

2-21 Start processing of sounding a hold tone during a call			
Classification	Voice communication service		
Function	Start processing of sounding a hold tone during a call		
Symbol	Elib_CV_Com_Start_To_Rev		
Syntax	int Elib_CV_Com_Start_To_Rev (ap_id);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Return value	Type	I/O	Description
Ret	Int	O	ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal
Include file	srv_cv.h		
Functional description	To sound a hold tone during a call, press the clear key. Call this function at the hold start.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.22 Stop processing of sound ing a hold tone during a call

2-22 Stop processing of sounding a hold tone during a call			
Classification	Voice communication service		
Function	Stop processing of sounding a hold tone during a call		
Symbol	Elib_CV_Com_Stop_To_Rev		
Syntax	int Elib_CV_Com_Stop_To_Rev (ap_id);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Return value	Type	I/O	Description
Ret	int	O	ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal
Include file	srv_cv.h		
Functional description	To stop sounding a hold tone during a call, press the clear key. Call this function at the hold stop.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.23 Processing of referring to the 64K communication/AV communication status

2-23 Processing of referring to the 64K communication/AV communication status			
Classification	Voice communication monitoring service		
Function	Processing of referring to the 64K communication/AV communication status		
Symbol	Elib_CV_UD_Conn_Status		
Syntax	int Elib_CV_UD_Conn_Status(void) ;		
Argument	Type	I/O	Description
None	void	-	-
Return value	Type	I/O	Description
status	int	-	Status ELIB_CV_UD_STOP : Under stop of 64K communication/AV communication ELIB_CV_UD_RUN : Under communication of 64K communication/AV communication ELIB_CV_UD_CALLED : Under incoming of 64K communication/AV communication ELIB_CV_UD_CALLING : Under outgoing of 64K communication/AV communication ELIB_CV_UD_DISCONNECT : Under disconnection of 64K communication/AV communication ELIB_CV_UD_CALLING_ALERTING : Under calling of 64K communication/AV communication ELIB_CV_UD_HOLD : Under hold of 64K communication/AV communication
Include file	srv_cv.h		
Functional description	Refers to the communication status of 64K communication/AV communication.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.24 Processing of referring to the internal/external AV communication status

2-24 Processing of referring to the internal/external AV communication status			
Classification	Voice communication monitoring service		
Function	Processing of referring to the internal/external AV communication status		
Symbol	Elib_CV_InOut_AV_Status		
Syntax	int Elib_CV_InOut_AV_Status(void) ;		
Argument	Type	I/O	Description
-	void	-	-
Return value	Type	I/O	Description
Ret	int	O	<p>Communication status</p> <p>ELIB_CV_AV_STOP : Under stop of AV communication</p> <p>ELIB_CV_In_AV_RUN : Under communication of internal AV communication</p> <p>ELIB_CV_In_AV_CALLED : Under incoming of internal AV communication</p> <p>ELIB_CV_In_AV_CALLING : Under outgoing of internal AV communication</p> <p>ELIB_CV_In_AV_DISCONNECT : Under disconnection of internal AV communication</p> <p>ELIB_CV_In_AV_CALLING_ALERTING : Start of calling the other party unit during internal outgoing</p> <p>ELIB_CV_In_AV_HOLD : Under hold of internal AV communication</p> <p>ELIB_CV_Out_AV_RUN : Under communication of external AV communication</p> <p>ELIB_CV_Out_AV_CALLED : Under incoming of external AV communication</p> <p>ELIB_CV_Out_AV_CALLING : Under outgoing of external AV communication</p> <p>ELIB_CV_Out_AV_DISCONNECT : Under disconnection of external AV communication</p> <p>ELIB_CV_Out_AV_CALLING_ALERTING : Start of calling the other party unit during external outgoing</p> <p>ELIB_CV_Out_AV_HOLD : Under hold of external AV communication</p>
Include file	Srv_cv.h		
Functional description	Refers to the communication status of internal/external AV communication.		
Related message			
Necessary procedure			
Note			
Prohibition			
Use example			

2.25 Processing of reading the communication status(with incoming type)

2-25 Processing of reading the communication status (with the incoming type)			
Classification	Voice communication service		
Function	Processing of reading the communication status (with the incoming type)		
Symbol	Elib_CV_Com_Status_Ref_Plus		
Syntax	Int Elib_CV_Com_Status_Ref_Plus (ap_id, rcv_scene_p);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
rcv_scene_p	Int *	O	Incoming type: ELIB_CV_RCV_SCENE_COMPETE_TRN : Outgoing convergence ELIB_CV_RCV_SCENE_RSV_RETURN : Incoming hold call ELIB_CV_RCV_SCENE_CALL_BACK : Re-incoming ELIB_CV_RCV_SCENE_NORMAL : Normal ELIB_CV_RCV_SCENE_NON : Unset
Return value	Type	I/O	Description
Status	int	O	Current communication status ELIB_CV_COM_STATUS_WAIT : Standby ELIB_CV_COM_STATUS_RCV : Under incoming ELIB_CV_COM_STATUS_TRN : Under outgoing ELIB_CV_COM_STATUS_DLV : Under calling ELIB_CV_COM_STATUS_TLK : Under conversation ELIB_CV_COM_STATUS_HLD : Under response hold ELIB_CV_COM_STATUS_RLS : Under release ELIB_CV_COM_STATUS_TLK_RCV : Under conversation and incoming ELIB_CV_COM_STATUS_TLK_TRN : Under conversation and outgoing ELIB_CV_COM_STATUS_TLK_DLV : Under conversation and calling ELIB_CV_COM_STATUS_TLK_RSV : Under conversation and hold ELIB_CV_COM_STATUS_RSV_RLS : Under hold and release ELIB_CV_COM_STATUS_TLK_RSV_RCV : Under conversation, hold, and incoming ELIB_CV_COM_STATUS_DUMMY1 : Under off-hook ELIB_CV_COM_STATUS_RCV_AV : Under incoming of an AV call ELIB_CV_COM_STATUS_TRN_AV : Under outgoing of an AV call ELIB_CV_COM_STATUS_DLV_AV :

			<p>Under calling of an AV call</p> <p>ELIB_CV_COM_STATUS_TLK_AV : </p> <p>Under conversation of an AV call</p> <p>ELIB_CV_COM_STATUS_HLD_AV : </p> <p>Under response hold of an AV call</p> <p>ELIB_CV_COM_STATUS_RLS_AV : Under AV release</p> <p>ELIB_CV_COM_STATUS_DUMMY2 : Under AV off-hook</p> <p>ELIB_CV_NG : Abnormal end</p>
Include file	Srv_cv.h		
Functional description	<p>Returns the communication status.</p> <p>Provides the incoming type only when the status is in the under incoming status or the under conversation and incoming status.</p>		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example	None		

2.26 Starting line status monitoring

2-26 Starting line status monitoring			
Classification	Line status monitoring service		
Function	Starting line status monitoring		
Symbol	Elib_CV_Start_Line_Watch		
Syntax	int Elib_CV_Start_Line_Watch (ap_id, mask, callbackFunc);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Mask	int	I	Mask of the notification event Starts notifying of an event with bit on. ELIB_CV_MASK_LINE_STATUS 0x01 : Line status change notification ELIB_CV_MASK_RESTRICT 0x02 : Restriction status change notification ELIB_CV_MASK_RSSI 0x08 : Receive level change notification ELIB_CV_MASK_ALL 0xff : All notified
CallbackFunc	MsbFunc	I	Callback function the event is notified
Return value	Type	I/O	Description
Sts	int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal
Include file	srv_cv.h		
Functional description	<p>Starts notifying of the event about the line status. Notifies of the following events to the requesting application.</p> <p>Events to be notified</p> <p><u>1. Line status change notification:</u></p> <ul style="list-style-type: none"> ● Notifies of that the line status is changed. ● The line status has the out-of-communication area status and the within-communication area status. ● Notifies of when within-communication area display primitive is received from DCF. <p><u>2. Restriction status change notification:</u></p> <ul style="list-style-type: none"> ● Notifies of when a restriction status change notification is received from DCF. <p><u>3. Receive level change notification:</u></p> <ul style="list-style-type: none"> ● Notifies of that the receive level is changed (three steps). <p><Line status change notification></p> <p>Line status change notification event structure</p> <pre>typedef struct { int category; VoiceNotify int subtype; VoiceNotify_AreaInfo int info; Line status }</pre>		


```

int      subinfo;  Line type
union {
...
} data ;          -> Unused
} ELIB_CV_EVENT ;

```

Line status

ELIB_CV_LINE_STATUS_OUT : Out-of-communication area
 ELIB_CV_LINE_STATUS_IN : Within-communication area

<Restriction status change notification>

Restriction status change notification event structure

```

typedef struct {
int      category; VoiceNotify
int      subtype;  VoiceNotify_Restrict
int      info;     Notification type
int      subinfo;  Restriction status
union {
...
    ELIB_CV_RES_CHG_INF  res_chg_inf ; -> Restriction display information
structure
} data ;
} ELIB_CV_EVENT ;

```

Notification type

ELIB_CV_RSMP_REST_STA : Restriction display start notification
 ELIB_CV_RSMP_REST_END : Restriction display end notification

Restriction status

The 0th bit is in the PS restriction status, and the 1st bit is in the CS restriction status.

ELIB_CV_BIT_RESTINF_CS 0x02 :

CS restriction information (Bit ON means "restricted." Bit OFF means "unrestricted.")

ELIB_CV_BIT_RESTINF_PS 0x01 :

PS restriction information (Bit ON means "restricted." Bit OFF means "unrestricted.")

The 2nd bit is in the PS emergency restriction status,
 and the 3rd bit is in the CS emergency restriction status."

ELIB_CV_BIT_ECRESTINF_CS 0x08 :

Emergency CS restriction information (Bit ON means "restricted." Bit OFF means "unrestricted.")

ELIB_CV_BIT_ECRESTINF_PS 0x04 :

Emergency PS restriction information (Bit ON means "restricted." Bit OFF means "unrestricted.")

	<p>Restriction display information</p> <pre>typedef struct tagELIB_CV_RES_CHG_INF { unsigned char NcRestriction; /* Normal transmission restriction */ unsigned char ServiceStatus; /* Service status */ unsigned char EcRestriction; /* Emergency transmission restriction */ } _ELIB_CV_RES_CHG_INF;</pre> <p>Normal transmission restriction and emergency transmission restriction</p> <pre>ELIB_CV_LINE_RESTRICT_DATA_ON /* With transmission restriction */ ELIB_CV_LINE_RESTRICT_DATA_OFF /* Without transmission restriction */</pre> <p>Service status</p> <pre>ELIB_CV_LINE_SRV_STATUS_CS /* Cs */ ELIB_CV_LINE_SRV_STATUS_PS /* Ps */ ELIB_CV_LINE_SRV_STATUS_CSPS /* Cs & Ps */</pre> <p><Receive level change notification></p> <p>Receive level change notification event structure</p> <pre>typedef struct { int category; VoiceNotify int subtype; VoiceNotify_RssiLevel int info; Receive level int subinfo; Line type union { ... } data; -> Unused } ELIB_CV_EVENT;</pre> <p>Receive level</p> <pre>ELIB_CV_RSSI_LEVEL_0 : Receive level 0 ELIB_CV_RSSI_LEVEL_1 : Receive level 1 ELIB_CV_RSSI_LEVEL_2 : Receive level 2 ELIB_CV_RSSI_LEVEL_3 : Receive level 3</pre>
Related message	None
Necessary procedure	None
Note	None
Prohibition	None
Use example	

2.27 Ending line status monitoring

2-27 Ending line status monitoring			
Classification	Line status monitoring service		
Function	Ending line status monitoring		
Symbol	Elib_CV_Stop_Line_Watch		
Syntax	int Elib_CV_Stop_Line_Watch (ap_id, mask);		
Argument	Type	I/O	Description
Ap_id	unsigned int	I	Application ID
mask	Int	I	Mask of the monitoring end event Ends notifying of an event with bit on. ELIB_CV_MASK_LINE_STATUS 0x01 : Line status change notification ELIB_CV_MASK_RESTRICT 0x02 : Restriction status change notification ELIB_CV_MASK_RSSI 0x08 : Receive level change notification ELIB_CV_MASK_ALL 0xff : All notified
Return value	Type	I/O	Description
Sts		O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal
Include file	srv_cv.h		
Functional description	Ends notifying on the event about the line status.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.28 Obtaining the receive level

2-28 Obtaining the receive level			
Classification	Line status monitoring service		
Function	Obtaining the receive level		
Symbol	Elib_CV_Get_RSSI		
Syntax	int Elib_CV_Get_RSSI (void);		
Argument	Type	I/O	Description
None	Void	-	-
Return value	Type	I/O	Description
level	Int	O	Receive level ELIB_CV_RSSI_LEVEL_0 : Receive level 0 ELIB_CV_RSSI_LEVEL_1 : Receive level 1 ELIB_CV_RSSI_LEVEL_2 : Receive level 2 ELIB_CV_RSSI_LEVEL_3 : Receive level 3
Include file	srv_cv.h		
Functional description	Obtains the current receive level. Even if the line status monitoring start function is not called for monitoring, only the status can be obtained by this function.		
Related message	None		
Necessary procedure	None		
Note			
Prohibition	None		
Use example			

2.29 Obtaining line status information

2-29 Obtaining line status information			
Classification	Line status monitoring service		
Function	Obtaining line status information		
Symbol	Elib_CV_Get_Line_Status		
Syntax	int Elib_CV_Get_Line_Status (net);		
Argument	Type	I/O	Description
Net	_ELIB_CV_AREAREF_CHG_INF *	O	Pointer of the area storing line status information
Return value	Type	I/O	Description
Sts	int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal
Include file	srv_cv.h		
Functional description	<p>Obtains the current receive level. Even if the line status monitoring start function is not called for monitoring, only the status can be obtained by this function.</p> <pre>typedef struct tagELIB_CV_AREAREF_CHG_INF { unsigned char LineStatus ; /* Within- and out-of-communication area status */ unsigned char CoverageStatus ; /* Area status information */ unsigned char RRcmode ; /* RRC mode */ unsigned char Network ; /* Network identification information */ unsigned char PlmnNumber[ELIB_CV_PLMN_LEN] ; /* PPLMN number */ unsigned char ServiceStatus_AREA ; /* CS and PS service status */ unsigned char RestrictStatus ; /* Restriction status */ unsigned char NcRestriction ; /* Normal transmission restriction */ unsigned char ServiceStatus_RES ; /* Service status */ unsigned char EcRestriction ; /* Emergency transmission restriction */ } _ELIB_CV_AREAREF_CHG_INF ;</pre> <p>Within- and out-of-communication area status ELIB_CV_LINE_STATUS_IN /* Within-communication area */ ELIB_CV_LINE_STATUS_OUT /* Out-of-communication area */</p> <p>Area status information ELIB_CV_LINE_CVR_STATUS_IN /* IN */ ELIB_CV_LINE_CVR_STATUS_OUT /* OUT */</p> <p>RRC mode ELIB_CV_LINE_RRC_MODE_IDLE /* idle-mode */ ELIB_CV_LINE_RRC_MODE_UTRAN /* utran-connected-mode */</p> <p>Network identification information ELIB_CV_LINE_NETWORK_HOME /* home */ ELIB_CV_LINE_NETWORK_VISIT /* visit */ ELIB_CV_LINE_NO_DATA /* No data */</p> <p>Service status</p>		

	<p> ELIB_CV_LINE_SRV_STATUS_CS /* Cs */ ELIB_CV_LINE_SRV_STATUS_PS /* Ps */ ELIB_CV_LINE_SRV_STATUS_CSPS /* Cs & Ps */ ELIB_CV_LINE_NO_DATA /* No data */ </p> <p>Restriction status</p> <p> ELIB_CV_LINE_RESTRICT_ON /* With transmission restriction */ ELIB_CV_LINE_RESTRICT_OFF /* Without transmission restriction */ </p> <p>Normal and emergency transmission restriction</p> <p> ELIB_CV_LINE_RESTRICT_DATA_ON /* With transmission restriction */ ELIB_CV_LINE_RESTRICT_DATA_OFF /* Without transmission restriction */ </p> <p>Service status</p> <p> ELIB_CV_LINE_SRV_STATUS_CS /* Cs */ ELIB_CV_LINE_SRV_STATUS_PS /* Ps */ ELIB_CV_LINE_SRV_STATUS_CSPS /* Cs & Ps */ </p>
Related message	None
Necessary procedure	None
Note	
Prohibition	None
Use example	

2.30 Obtaining the within- or out-of-communication area status

2-30 Obtaining the within- or out-of-communication area status				
Classification		Linse status monitoring service		
Function		Obtaining the within- or out-of-communication area information		
Symbol		Elib_CV_Get_Line_Status_Ex		
Syntax		int Elib_CV_Get_Line_Status_Ex (status_p);		
Argument		Type	I/O	Description
Status_p		_ELIB_CV_LINE_STATUS_EX *		Pointer of the area storing the information of the within- and out-of-communication area statuses
	area_sts	unsigned char	O	Within- or out-of communication area status ELIB_CV_LINE_STATUS_IN : Within-communication area ELIB_CV_LINE_STATUS_OUT : Out-of-communication area
	Wlan	unsigned char	O	Unused
Return value		Type	I/O	Description
Sts		int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal
Include file		srv_cv.h		
Functional description		Obtains the information on the current status of the within- and out-of-communication areas for each line.		
Related message		None		
Necessary procedure		None		
Note		None		
Prohibition		None		
Use example				

2.31 PLMN Number information obtainment function

2-31 PLMN Number information obtainment function																																		
Classification	Linse status monitoring service																																	
Function	PLMN number informaiton obtainment function																																	
Symbol	Elib_CV_Get_PLMNNumber																																	
Syntax	void Elib_CV_Get_PLMNNumber (_ELIB_CV_PLMN_DATA *PLMNNumber_data) ;																																	
Argument	Type	I/O	Description																															
PLMNNumber_Data	_ELIB_CV_PLMN_DATA*	O	Pointer of the area storing line status information																															
Return value	Type	I/O	Description																															
-	Void	-	-																															
Include file	SRV_NW.H																																	
Functional description	<p>Obtains PLMN number information from information elements of AreaStatus-ind (within-communication area display).</p> <p>Structure for storing PLMNNumber data</p> <pre>typedef struct tagELIB_CV_PLMN_DATA { unsigned char PLMNNumber_data[ELIB_CV_PLMN_LEN]; /* PLMNNumber information */ }_ELIB_CV_PLMN_DATA</pre> <pre>#define ELIB_CV_PLMN_LEN 3 /* Arrangement number of stored PLMN numbers */</pre> <p>For voice communication service, information is stored as follows:</p> <pre>PLMNNumber_data[0] = 0x44 MCC information PLMNNumber_data[1] = 0xF0 MCC information PLMNNumber_data[2] = 0x01 MNC information</pre>																																	
	<p>PLMNNumber information included in AreaStatus_Ind comprises three bytes. 12 bits from the beginning contains MCC information, the next four bits is used for MCC expansion, and the last eight bits contains MNC information.</p> <p>Supplementation</p> <table><tr><td></td><td>Higher-order bit</td><td>Lower-order bit</td><td></td></tr><tr><td>Memory address</td><td colspan="2">+-----+-----+</td><td></td></tr><tr><td>0</td><td> MCC</td><td> </td><td>0x44</td></tr><tr><td></td><td colspan="2">+-----+</td><td>+</td></tr><tr><td>1</td><td> Expansion</td><td> </td><td>0xF0</td></tr><tr><td></td><td colspan="2">+-----+</td><td>+</td></tr><tr><td>2</td><td> MNC</td><td> </td><td>0x01</td></tr><tr><td></td><td colspan="2">+-----+</td><td>+</td></tr></table> <p>- For test SIM, MCC = 0x001, and MNC = 0x01.</p> <pre>PLMNNumber_data[0] = 0x00</pre>				Higher-order bit	Lower-order bit		Memory address	+-----+-----+			0	MCC		0x44		+-----+		+	1	Expansion		0xF0		+-----+		+	2	MNC		0x01		+-----+	
	Higher-order bit	Lower-order bit																																
Memory address	+-----+-----+																																	
0	MCC		0x44																															
	+-----+		+																															
1	Expansion		0xF0																															
	+-----+		+																															
2	MNC		0x01																															
	+-----+		+																															

	PLMNNumber_data[1] = 0xF1 PLMNNumber_data[2] = 0x10 <ul style="list-style-type: none"> - For MCC expansion, always F is stored because the area is not used currently. - For the out-of-communication area status, 1 is placed in all bits. If AreaStatus_Ind is not obtained, 0 is placed in all bits. - During out-of-communication AreaStatus_Ind is not obtained PLMNNumber_data[0] = 0xFF PLMNNumber_data[0] = 0x00 PLMNNumber_data[1] = 0xFF PLMNNumber_data[1] = 0x00 PLMNNumber_data[2] = 0xFF PLMNNumber_data[2] = 0x00
Related message	None
Necessary procedure	None
Note	None
Prohibition	None
Use example	

2.32 Obtaining information on stored answering messages

2-32 Obtaining information on stored answering messages			
Classification	Network service		
Function	Obtaining information on stored answering messages		
Symbol	Elib_CV_Get_VM_Stored		
Syntax	int Elib_CV_Get_VM_Stored (vm_num_p);		
Argument	Type	I/O	Description
vm_num_p	int *	O	Address of the storage area of the number of stored answering messages
Return value	Type	I/O	Description
Sts	int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal (failure of nonvolatile obtainment)
Include file	srv_cv.h		
Functional description	Obtains the storage status of answering messages. When the processing result is ELIB_CV_OK, the number of stored answering messages is stored in vm_num_p.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.33 Setting the number of answering messages to be stored

2-33 Setting the number of answering messages to be stored			
Classification	Network service		
Function	Setting the number of answering messages to be stored		
Symbol	Elib_CV_Set_VM_Stored		
Syntax	int Elib_CV_Set_VM_Stored (vm_num) ;		
Argument	Type	I/O	Description
Vm_num	int	I	The number of stored answering messages
Return value	Type	I/O	Description
Sts	int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal (failure of nonvolatile setting)
Include file	srv_cv.h		
Functional description	Sets the storage status of answering messages to nonvolatile.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.34 N + 1 Obtaining selection of the incoming function – getting of the incoming function

2-34 N + 1 Obtaining selection of the incoming function - getting of the incoming function			
Classification	Network service		
Function	Obtaining selection of the incoming function - getting of the incoming function		
Symbol	Elib_CV_Get_RcvCall_Select		
Syntax	int Elib_CV_Get_RcvCall_Select (void) ;		
Argument	Type	I/O	Description
None	void	-	
Return value	Type	I/O	Description
Ret	int	O	ELIB_CV_TYAKUSIN_RUSUBAN : Answer ELIB_CV_TYAKUSIN_TENSOU : Forward ELIB_CV_TYAKUSIN_KYOHI : Reject (disconnect) ELIB_CV_TYAKUSIN_TUJOU : RECEIPT OF AN INCOMING CALL (NORMAL INCOMING)
Include file	srv_cv.h		
Functional description	Obtains the settings of selection of the incoming function - getting of the incoming function from SRAM.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.35 N + 1 Setting selection of the incoming function

2-35 N+ 1 Setting selection of the incoming function - setting of the incoming function			
Classification	Network service		
Function	Setting selection of incoming call function - setting of the incoming function		
Symbol	Elib_CV_Set_RcvCall_Select		
Syntax	int Elib_CV_Set_RcvCall_Select (func);		
Argument	Type	I/O	Description
Func	int	I	ELIB_CV_TYAKUSIN_RUSUBAN : Answer ELIB_CV_TYAKUSIN_TENSOU : Forward ELIB_CV_TYAKUSIN_KYOHI : Reject (disconnect) ELIB_CV_TYAKUSIN_TUJOU : Receipt of an incoming call (normal incoming call)
Return value	Type	I/O	Description
Sts	int	O	Processing result ELIB_CV_OK : NORMAL ELIB_CV_NG : Abnormal
Include file	srv_cv.h		
Functional description	Sets the settings of selection of the incoming function - setting of the incoming function to SRAM.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.36 Registering additional service setting

2-36 Registering additional service setting				
Classification		Network service		
Function		Registering additional service setting		
Symbol		Elib_CV_Set_AddSrvInfo		
Syntax		int Elib_CV_Set_AddSrvInfo (data_no, add_data_p);		
Argument		Type	I/O	Description
Data_no		int	I	Registration number: 1 to 10
add_data_p		_ELIB_CV_ADDSRV_D ATA *	I	Pointer of additional service setting data
	Flg	unsigned char		Identifier flag ELIB_CV_NO_FLG : No data ELIB_CV_OPT_FLG : Special number ELIB_CV_USSD_FLG : USSD
	Title[ELIB_SRVINFO_TITLE]	char		Service setting name #define ELIB_SRVINFO_TITLE 21 /* The 21st byte is the NULL character */
	Send_No[ELIB_SRVINFO_SENDNO]	char		Dial data #define ELIB_SRVINFO_SENDNO 40
Return value		Type	I/O	Description
Sts		int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal (registration failure)
Include file		srv_cv.h		
Functional description		Registers additional service settings (send information, a service setting name, and a special number or USSD) to the registration number (SRAM) specified by the argument.		
Related message		None		
Necessary procedure		None		
Note		* Even if setting is overwritten to the registration number that has already been registered, normal (OK) is returned. * Data that becomes discontinuous by registration or deletion is not relocated or sorted. An application program should relocate or sort data. * As a service setting name, up to 20 bytes can be registered.		
Prohibition		None		
Use example				

2.37 Obtaining additional service setting

2-37 Obtaining additional service setting				
Classification		Network service		
Function		Obtaining additional service setting		
Symbol		Elib_CV_Get_AddSrvInfo		
Syntax		int Elib_CV_Get_AddSrvInfo (data_no, add_data_p);		
Argument		Type	I/O	Description
Data_no		int	I	Registration number : 1 to 10
add_data_p		ELIB_CV_ADDSRV_DATA *	O	Pointer of additional service setting data
	Flg	unsigned char		Identifier flag ELIB_CV_NO_FLG : No data ELIB_CV_OPT_FLG : Special number ELIB_CV_USSD_FLG : USSD
	Title [ELIB_SRVINFO_TITLE]	char		Service setting name #define ELIB_SRVINFO_TITLE 21 /* The 21st byte is the NULL character */
	Send_No[ELIB_SRVINFO_SENDNO]	char		Dial data #define ELIB_SRVINFO_SENDNO 40
Return value		Type	I/O	Description
Sts		int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal (obtainment failure)
Include file		srv_cv.h		
Functional description		Obtains additional service settings (send information, a service setting name, and a special number or USSD) that are stored in the registration number specified by the argument from SRAM.		
Related message		None		
Necessary procedure		None		
Note		* Data that becomes discontinuous by registration or deletion is not relocated or sorted. An application program should relocate or sort data.		
Prohibition		None		
Use example				

2.38 Deleting additional service setting

2-38 Deleting additional service setting			
Classification	Network service		
Function	Deleting additional service setting		
Symbol	Elib_CV_Del_AddSrvInfo		
Syntax	int Elib_CV_Del_AddSrvInfo (data_no) ;		
Argument	Type	I/O	Description
Data_no	int	I	Registration number : 1 to 10
Return value	Type	I/O	Description
sts	int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal (deletion failure)
Include file	srv_cv.h		
Functional description	Deletes the data of additional service settings (send information, a service setting name, a special number or USSD, and a registration number) of the specified registration number.		
Related message	None		
Necessary procedure	None		
Note	* Data that becomes discontinuous by registration or deletion is not relocated or sorted. An application program should relocate or sort data.		
Prohibition	None		
Use example			

2.39 Deleting all additional service setting

2-39 Deleting all additional service setting			
Classification	Network service		
Function	Deleting all additional service setting		
Symbol	Elib_CV_DelAll_AddSrvInfo		
Syntax	int Elib_CV_DelAll_AddSrvInfo (void) ;		
Argument	Type	I/O	Description
None	void	-	
Return value	Type	I/O	Description
sts	int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal (failure of all deletion)
Include file	srv_cv.h		
Functional description	Deletes the data of all additional service settings (send information, a service setting name, a special number or USSD, and a registration number) from SRAM.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.40 Registering response message setting

2-40 Registering response message setting			
Classification	Network service function		
Function	Registering response message setting		
Symbol	Elib_CV_Set_AddResponseMsg		
Syntax	int Elib_CV_Set_AddResponseMsg (data_no, add_data_p);		
Argument	Type	I/O	Description
data_no	unsigned char	I	Registration number (0 to 9)
add_data_p	_ELIB_CV_RESPONSE_MSG_DATA*	I	Pointer to the additional response message setting data area
Return value	Type	I/O	Description
Ret	int	O	ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal (setting failure)
Include file	srv_cv.h		
Functional description	<p>Registers additional response service settings (send information, a service setting name, and USSD) to the registration number specified by the argument.</p> <p>* A service setting name and the data area of dial data have 20 bytes. (The 21st byte is the end code.)</p> <pre>typedef struct tagELIB_CV_RESPONSE_MSG_DATA { unsigned char Title[ELIB_RESMSG_TITLE] ; /* Service setting name */ unsigned char Rcv_Msg[ELIB_RESMSG_DATA]; /* Dial data */ } _ELIB_CV_RESPONSE_MSG_DATA ;</pre> <pre>#define ELIB_RESMSG_TITLE MSL_SRVRVCVMSG_TITLE+1 /* Response message title maximum length 21 */ #define ELIB_RESMSG_DATA MSL_SRVRVCVMSG_DATA+1 /* Response message data maximum length 21 */</pre>		
Related message			
Necessary procedure			
Note	<ul style="list-style-type: none"> - Even if data is overwritten to the registration number that has already been registered, normal (OK) is returned. - Data that becomes discontinuous by registration or deletion is not relocated nor sorted. <p>An application should relocated or sort data.</p>		
Prohibition			
Use example			

2.41 Registering response message setting

2-41 Obtaining response message setting			
Classification	Network service function		
Function	Obtaining response message setting		
Symbol	Elib_CV_Get_AddResponseMsg		
Syntax	int Elib_CV_Get_AddResponseMsg (data_no, add_data_p);		
Argument	Type	I/O	Description
Data_no	unsigned char	I	Registration number (0 to 9)
add_data_p	_ELIB_CV_RESPONSE_MSG_DATA *	O	Pointer to the additional response message setting data area
Return value	Type	I/O	Description
Ret	Int	O	ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal (setting failure)
Include file	srv_cv.h		
Functional description	<p>Obtains additional response service settings (send information, a service setting name, and USSD) from the registration number specified by the argument.</p> <p>A service setting name and the data area of dial data have 20 bytes. (The 21st byte is the end code.)</p> <pre>typedef struct tagELIB_CV_RESPONSE_MSG_DATA { unsigned char Title[ELIB_RESMSG_TITLE] ; /* Service setting name */ unsigned char Rcv_Msg[ELIB_RESMSG_DATA]; /* Dial data */ } _ELIB_CV_RESPONSE_MSG_DATA ;</pre> <pre>#define ELIB_RESMSG_TITLE MSL_SRVRVMSG_TITLE+1 /* Response message title maximum length 21 */ #define ELIB_RESMSG_DATA MSL_SRVRVMSG_DATA+1 /* Response message data maximum length 21 */</pre>		
Related message			
Necessary procedure			
Note	<p>Data that becomes discontinuous by registration or deletion is not relocated nor sorted.</p> <p>An application should relocated or sort data.</p>		
Prohibition			
Use example			

2.42 Deleting response message setting

2-42 Deleting response message setting			
Classification	Network service function		
Function	Deleting response message setting		
Symbol	Elib_CV_Del_AddResponseMsg		
Syntax	int Elib_CV_Del_AddResponseMsg (data_no) ;		
Argument	Type	I/O	Description
data_no	unsigned char	I	Registration number (0 to 9)
Return value	Type	I/O	Description
ret	Int	O	ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal (setting failure)
Include file	srv_cv.h		
Functional description	Deletes additional response service settings (send information, a service setting name, and USSD) from the registration number specified by the argument.		
Related message			
Necessary procedure			
Note	Data that becomes discontinuous by registration or deletion is not relocated nor sorted. An application should relocated or sort data.		
Prohibition			
Use example			

2.43 Deleting all response message setting

2-43 Deleting all response message setting			
Classification	Network service function		
Function	Deleting all response message setting		
Symbol	Elib_CV_DelAll_AddResponseMsg		
Syntax	int Elib_CV_DelAll_AddResponseMsg (void) ;		
Argument	Type	I/O	Description
-	void	-	-
Return value	Type	I/O	Description
Ret	int	O	ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal (setting failure)
Include file	SRV_CV.H		
Functional description	Deletes data of all response service settings (send information, a service setting name, USSD, and a registration number).		
Related message			
Necessary procedure			
Note			
Prohibition			
Use example			

2.44 Requesting reconnection tone setting

2-44 Requesting reconnection tone setting			
Classification	Other services		
Function	Requesting reconnection tone setting		
Symbol	Elib_CV_Req_Reconn_Ctrl		
Syntax	int Elib_CV_Req_Reconn_Ctrl(reconn);		
Argument	Type	I/O	Description
reconn	int	I	Reconnection tone setting ELIB_CV_RECONN_ON_T_OFF : (Tone OFF) ELIB_CV_RECONN_ON_T_LOW : (Tone ON low tone) ELIB_CV_RECONN_ON_T_HI : (Tone ON high tone)
Return value	Type	I/O	Description
sts	int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal
Include file	srv_cv.h		
Functional description	Performs only the nonvolatile setting of the reconnection tone. The processing of requesting ON or OFF of the reconnection setting is not performed.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.45 Obtaining reconnection tone setting

2-45 Obtaining reconnection tone setting			
Classification	Other services		
Function	Obtaining reconnection tone setting		
Symbol	Elib_CV_Get_Reconn		
Syntax	int Elib_CV_Get_Reconn (void);		
Argument	Type	I/O	Description
None	void	-	
Return value	Type	I/O	Description
Recon	int	O	Reconnection tone setting ELIB_CV_RECONN_ON_T_OFF : (Tone OFF) ELIB_CV_RECONN_ON_T_LOW : (Tone ON low tone) ELIB_CV_RECONN_ON_T_HI : (Tone ON high tone)
Include file	srv_cv.h		
Functional description	Obtains reconnection tone setting.		
Related message	None		
Necessary procedure	None		

Note	None
Prohibition	None
Use example	

2.46 Processing of obtaining the noise canceller setting status

2-46 Processing of obtaining the noise canceller setting status			
Classification	Other services		
Function	Processing of obtaining the noise canceller setting status		
Symbol	Elib_CV_Get_Noise_Cancel		
Syntax	int Elib_CV_Get_Noise_Cancel(void);		
Argument	Type	I/O	Description
None	void	-	-
Return value	Type	I/O	Description
reconn	int	O	Noise canceller setting ELIB_CV_ON : Noise canceller ON ELIB_CV_OFF: Noise canceller OFF
Include file	srv_cv.h		
Functional description	Obtains the setting status of noise canceller.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.47 Processing of noise canceller setting

2-47 Processing of noise canceller setting			
Classification	Other services		
Function	Processing of noise canceller setting		
Symbol	Elib_CV_Set_Noise_Cancel		
Syntax	Int Elib_CV_Set_Noise_Cancel(mode);		
Argument	Type	I/O	Description
mode	int	I	Noise canceller setting ELIB_CV_ON : Noise canceller ON ELIB_CV_OFF: Noise canceller OFF
Return value	Type	I/O	Description
Sts	int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal
Include file	srv_cv.h		
Functional description	Sets noise canceller.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		

Use example	
-------------	--

2.48 Processing of noise canceller setting

2-48 Processing of obtaining the call quality alarm setting status			
Classification	Other services		
Function	Processing of obtaining the call quality alarm setting status		
Symbol	Elib_CV_Get_Quality_Inferior_Alm		
Syntax	int Elib_CV_Get_Quality_Inferior_Alm(void) ;		
Argument	Type	I/O	Description
None	void		-
Return value	Type	I/O	Description
mode	int	O	ELIB_CV_QUALITY_ALM_OFF: Quality alarm OFF ELIB_CV_QUALITY_ALM_LOW: Quality alarm ON low tone ELIB_CV_QUALITY_ALM_HI : Quality alarm ON high tone
Include file	srv_cv.h		
Functional description	Obtains the setting status of the call quality alarm.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.49 Processing of call quality alarm setting

2-49 Processing of call quality alarm setting			
Classification	Other services		
Function	Processing of call quality alarm setting		
Symbol	Elib_CV_Set_Quality_Inferior_Alm		
Syntax	int Elib_CV_Set_Quality_Inferior_Alm(mode) ;		
Argument	Type	I/O	Description
Mode	int	I	ELIB_CV_QUALITY_ALM_OFF: Quality alarm OFF ELIB_CV_QUALITY_ALM_LOW: Quality alarm ON low tone ELIB_CV_QUALITY_ALM_HI : Quality alarm ON high tone
Return value	Type	I/O	Description
Sts	int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal
Include file	SRV_CV.H		
Functional description	Sets the call quality alarm.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		

Use example	
-------------	--

2.50 Processing of obtaining noise canceller permission/non-permission

2-50 Processing of obtaining noise canceller permission/non-permission			
Classification	Other services		
Function	Processing of obtaining noise canceller permission/non-permission		
Symbol	Elib_CV_Get_Noise_Cancel_Permit_Check		
Syntax	int Elib_CV_Get_Noise_Cancel_Permit_Check(void);		
Argument	Type	I/O	Description
None	void	-	-
Return value	Type	I/O	Description
Sts	int	O	Processing result ELIB_CV_ON : Noise canceller permission ELIB_CV_OFF : NOISE CANCELLER NON-PERMISSION
Include file	srv_cv.h		
Functional description	Obtains whether noise canceller is permitted.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.51 Processing of referring to the radio unit status

2-51 Processing of referring to the radio unit status			
Classification	Other services		
Function	Processing of referring to the radio unit status		
Symbol	Elib_CV_RF_Status		
Syntax	int Elib_CV_RF_Status (void);		
Argument	Type	I/O	Description
None	void		-
Return value	Type	I/O	Description
Sts	int	O	Status ELIB_CV_RF_STOP: The radio unit is being stopped. ELIB_CV_RF_RUN : The radio unit is operating.
Include file	SRV_CV.H		
Functional description	Refers to the radio unit status.		
Related message	None		
Necessary procedure	None		
Note	The status obtained by this function does not always mean that stop or start the radio unit is completed. The status is changed when a stop or start request is sent to an lower side. Therefore, the radio unit may be in the middle of stop or		

	start processing.
Prohibition	None
Use example	

2.52 Processing of obtaining the priority communication mode setting state

2-52 Processing of obtaining the priority communication mode setting status			
Classification	Other services		
Function	Processing of obtaining the priority communication mode setting status		
Symbol	Elib_CV_Get_ComPriority_Mode		
Syntax	int Elib_CV_Get_ComPriority_Mode (void) ;		
Argument	Type	I/O	Description
None	void	-	-
Return value	Type	I/O	Description
mode	int	O	ELIB_CV_COMPRI_NONE : No setting ELIB_CV_COMPRI_VOICE : Voice ELIB_CV_COMPRI_PACKET: Packet
Include file	srv_cv.h		
Functional description	Obtains the setting status of the priority communication mode.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.53 Processing of priority communication mode setting

2-53 Processing of priority communication mode setting			
Classification	Other services		
Function	Processing of priority communication mode setting		
Symbol	Elib_CV_Set_ComPriority_Mode		
Syntax	int Elib_CV_Set_ComPriority_Mode (mode) ;		
Argument	Type	I/O	Description
mode	int	-	ELIB_CV_COMPRI_NONE : No setting ELIB_CV_COMPRI_VOICE : Voice ELIB_CV_COMPRI_PACKET: Packet
Return value	Type	I/O	Description
Sts	int		Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal
Include file	srv_cv.h		
Functional description	Sets the priority communication mode.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		

Use example	
-------------	--

2.54 Obtainig voice message sound setting

2-54 Obtaining voice message sound setting			
Classification	Other services		
Function	Obtaining voice message sound setting		
Symbol	Elib_CV_Get_VM_Runbling		
Syntax	int Elib_CV_Get_VM_Runbling (void) ;		
Argument	Type	I/O	Description
None	void	-	-
Return value	Type	I/O	Description
Mode	int	O	Sound setting ELIB_CV_ON : Message sound ON ELIB_CV_OFF : Message sound OFF
Include file	srv_cv.h		
Functional description	Obtains the setting whether the phone sounds as the voice message service when the number of messages is increased.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.55 Voice message sound setting

2-55 Voice message sound setting			
Classification	Other services		
Function	Voice message sound setting		
Symbol	Elib_CV_Set_VM_Runbling		
Syntax	int Elib_CV_Set_VM_Runbling (mode) ;		
Argument	Type	I/O	Description
mode	int	I	Sound setting ELIB_CV_ON : Message sound ON ELIB_CV_OFF : Message sound OFF
Return value	Type	I/O	Description
Sts	int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal
Include file	srv_cv.h		
Functional description	Sets whether the phone sounds as the voice message service when the number of messages is increased.		
Related message	None		
Necessary procedure	None		

CE Linux Forum Technical Document

Note	None
Prohibition	None
Use exam ple	

2.56 Processing of obtaining the setting status of automatic incoming ON/OFF

2-56 Processing of obtaining the setting status of automatic incoming ON/OFF			
Classification	Other services		
Function	Processing of obtaining the setting status of automatic incoming ON/OFF		
Symbol	Elib_CV_Get_Auto_Receive_Mode		
Syntax	int Elib_CV_Get_Auto_Receive_Mode(void);		
Argument	Type	I/O	Description
None	void	-	
Return value	Type	I/O	Description
Sts	int	O	Setting status ELIB_CV_ON : Automatic incoming ON ELIB_CV_OFF : Automatic incoming OFF
Include file	srv_cv.h		
Functional description	Obtains the automatic incoming ON/OFF status.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.57 Automatic incoming ON/OFF setting processing

2-57 Automatic incoming ON/OFF setting processing			
Classification	Other services		
Function	Automatic incoming ON/OFF setting processing		
Symbol	Elib_CV_Set_Auto_Receive_Mode		
Syntax	int Elib_CV_Set_Auto_Receive_Mode(mode);		
Argument	Type	I/O	Description
mode	int	I	Setting status ELIB_CV_ON : Automatic incoming ON ELIB_CV_OFF : Automatic incoming OFF
Return value	Type	I/O	Description
Sts	int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal
Include file	srv_cv.h		
Functional description	Sets the automatic incoming ON/OFF status.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		

Use example	
-------------	--

2.58 Processing of obtaining the setting status of the automatic incoming timer value

2-58 Processing of obtaining the setting status of the automatic incoming timer value			
Classification	Other services		
Function	Processing of obtaining the setting status of the automatic incoming timer value		
Symbol	Elib_CV_Get_Auto_Receive_Timer		
Syntax	int Elib_CV_Get_Auto_Receive_Timer (void);		
Argument	Type	I/O	Description
None	void	-	-
Return value	Type	I/O	Description
timer	int	O	Automatic incoming timer value 1 to 120 (seconds)
Include file	srv_cv.h		
Functional description	Obtains the automatic incoming timer value.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.59 Processing of automatic incoming timer value setting

2-59 Processing of automatic incoming timer value setting			
Classification	Other services		
Function	Processing of automatic incoming timer value setting		
Symbol	Elib_CV_Set_Auto_Receive_Timer		
Syntax	int Elib_CV_Set_Auto_Receive_Timer (timer);		
Argument	Type	I/O	Description
timer		I	Automatic incoming timer value 1 to 120 (seconds)
Return value	Type	I/O	Description
Sts		O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal
Include file	srv_cv.h		
Functional description	Sets the automatic incoming timer value.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.60 Obtaining the totalized reset data

2-60 Obtaining the totalized reset data				
Classification		Other services		
Function		Obtaining the totalized reset data		
Symbol		Elib_CV_Get_Clr_Time		
Syntax		int Elib_CV_Get_Clr_Time (reset_time);		
Argument		Type	I/O	Description
reset_time		_ELIB_CV_RSTDATE *	O	Address of the area storing the totalized reset data
	Month	unsigned char	O	Month
	Day	unsigned char	O	Day
	Hour	unsigned char	O	Hour
	Min	unsigned char	O	Minute
Return value		Type	I/O	Description
sts		Int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : ABNORMAL
Include file		srv_cv.h		
Functional description		Obtains the totalized reset date. Obtains the information on the totalized reset date from non-volatility.		
Related message		None		
Necessary procedure		None		
Note		None		
Prohibition		None		
Use example				

2.61 Setting the totalized reset date

2-61 Setting the totalized reset date			
Classification	Other services		
Function	Setting the totalized reset date		
Symbol	Elib_CV_Set_Clr_Time		
Syntax	int Elib_CV_Set_Clr_Time (void);		
Argument	Type	I/O	Description
None	void	-	-
Return value	Type	I/O	Description
Sts	int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : ABNORMAL
Include file	srv_cv.h		
Functional description	Sets the totalized reset date. Writes the totalized reset date to non-volatility. If the date is set to OFF, ELIB_CV_NG is returned.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.62 Setting inter-application transmittance event monitoring

2-62 Starting inter-application transmittance event monitoring			
Classification	Other services		
Function	Starting inter-application transmittance event monitoring		
Symbol	Elib_CV_Start_Trans_Event_Watch		
Syntax	int Elib_CV_Start_Trans_Event_Watch (ap_id, mask, callbackFunc);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Mask	unsigned int	I	<p>Mask of the notification event</p> <p>Sets the bit corresponding to the event requesting the notification to ON.</p> <p>To request multiple event notifications, set each bit to OR.</p> <p>ELIB_CV_MASK_TRANS_EVENT_01 0x01: Event 1 ELIB_CV_MASK_TRANS_EVENT_02 0x02: Event 2 ELIB_CV_MASK_TRANS_EVENT_03 0x04: Event 3 ELIB_CV_MASK_TRANS_EVENT_04 0x08: Event 4 ELIB_CV_MASK_TRANS_EVENT_05 0x10: Event 5 ELIB_CV_MASK_TRANS_EVENT_06 0x20: Event 6 ELIB_CV_MASK_TRANS_EVENT_07 0x40: Event 7 ELIB_CV_MASK_TRANS_EVENT_08 0x80: Event 8 ELIB_CV_MASK_ALL 0xff: All notified</p>
CallbackFunc	MsbFunc	I	<p>Callback function the event is notified</p> <p>If the same application tries to register different callback functions for the same event, the callback function registered last is effective.</p>
Return value	Type	I/O	Description
Sts	int	O	<p>Processing result</p> <p>ELIB_CV_OK: Normal ELIB_CV_NG: Abnormal</p>
Include file	srv_cv.h		
Functional description	<p>An application uses "Requesting a notification of an inter-application transmittance event" to start broadcasting an event notification to applications.</p> <p>Use this function to exchange events related to any CS service between applications.</p> <p><Events to be notified></p> <p><u>Inter-application transmittance event notification:</u></p> <ul style="list-style-type: none"> Reports the event set in "Requesting a notification of an inter-application transmittance event" transparently. <p>Structure of inter-application transmittance event notification</p> <pre>typedef struct { int category; -> VoiceNotify int subtype; -> Notification event int info; -> Value set by the application the event is notified</pre>		

	<pre> int subinfo; -> Value set by the application the event is notified union { unsigned char data[ELIB_CV_TRAN_DATA_SIZE]; -> Value set by the application the event is notified } data ; } _ELIB_CV_EVENT ; #define ELIB_CV_TRAN_DATA_SIZE 256 /* Data size of the inter-application transmittance event*/ Notification event: VoiceNotify_TranEvent_01 /* Event 1 */ VoiceNotify_TranEvent_02 /* Event 2 */ VoiceNotify_TranEvent_03 /* Event 3 */ VoiceNotify_TranEvent_04 /* Event 4 */ VoiceNotify_TranEvent_05 /* Event 5 */ VoiceNotify_TranEvent_06 /* Event 6 */ VoiceNotify_TranEvent_07 /* Event 7 */ VoiceNotify_TranEvent_08 /* Event 8 */ </pre>
Related message	Inter-application transmittance event notification
Necessary procedure	None
Note	<ul style="list-style-type: none"> - The notification event can receive up to eight events determined arbitrarily between applications. - How to identify notification events needs to be determined arbitrarily between applications. - The CS service has no concern in the data notified as an event, because the any value set by the sending application is notified.
Prohibition	<p>The events that are not related the CS service is prohibited from being sent and received.</p> <p>Even if an event that is not related to the CS service is received, the operation is not secured.</p>
Use example	

2.63 Ending inter-application transmittance event monitoring

2-63 Ending inter-application transmittance event monitoring			
Classification	Other services		
Function	Ending inter-application transmittance event monitoring		
Symbol	Elib_CV_Stop_Tran_Event_Watch		
Syntax	int Elib_CV_Stop_Tran_Event_Watch (ap_id, mask);		
Argument	Type	I/O	Description
Ap_id	unsigned int	I	Application ID
Mask	unsigned int	I	<p>Mask of the notification event</p> <p>Sets the bit corresponding to the event ending the notification to ON.</p> <p>To end multiple event notifications, set each bit to OR.</p> <p>ELIB_CV_MASK_TRANS_EVENT_01 0x01 : Event 1</p> <p>ELIB_CV_MASK_TRANS_EVENT_02 0x02 : Event 2</p> <p>ELIB_CV_MASK_TRANS_EVENT_03 0x04 : Event 3</p> <p>ELIB_CV_MASK_TRANS_EVENT_04 0x08 : Event 4</p> <p>ELIB_CV_MASK_TRANS_EVENT_05 0x10 : Event 5</p> <p>ELIB_CV_MASK_TRANS_EVENT_06 0x20 : Event 6</p> <p>ELIB_CV_MASK_TRANS_EVENT_07 0x40 : Event 7</p> <p>ELIB_CV_MASK_TRANS_EVENT_08 0x80 : Event 8</p> <p>ELIB_CV_MASK_ALL 0xff : All notified</p>
Return value	Type	I/O	Description
sts			<p>Processing result</p> <p>ELIB_CV_OK : Normal</p> <p>ELIB_CV_NG : Abnormal</p>
Include file	srv_cv.h		
Functional description	<p>An application uses "Requesting a notification of an inter-application transmittance event" to end broadcasting an event notification to applications.</p> <p>For notifying of an event, see "Starting inter-application transmittance event monitoring".</p>		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.64 Requesting a notification of an inter-application transmittance event

2-64 Requesting a notification of an inter-application transmittance event			
Classification	Other services		
Function	Requesting a notification of an inter-application transmittance event		
Symbol	Elib_CV_Trans_Event_Broadcast		
Syntax	int Elib_CV_Trans_Event_Broadcast (type, info, subinfo, data) ;		
Argument	Type	I/O	Description
Type	int	I	<p>Event type</p> <p>Specify one of the following types</p> <p>VoiceNotify_TransEvent_01 : Event 1</p> <p>VoiceNotify_TransEvent_02 : Event 2</p> <p>VoiceNotify_TransEvent_03 : Event 3</p> <p>VoiceNotify_TransEvent_04 : Event 4</p> <p>VoiceNotify_TransEvent_05 : Event 5</p> <p>VoiceNotify_TransEvent_06 : Event 6</p> <p>VoiceNotify_TransEvent_07 : Event 7</p> <p>VoiceNotify_TransEvent_08 : Event 8</p>
Info	int	I	<p>Option</p> <p>The value is set to info in the structure of the inter-application transmittance event notification.</p>
Subinfo	int	I	<p>Option</p> <p>The value is set to subinfo in the structure of the inter-application transmittance event notification.</p>
Data	void *	I	<p>Option</p> <p>The value is set to data in the structure of the inter-application transmittance event notification.</p>
Return value	Type	I/O	Description
Sts	Int	O	<p>Processing result</p> <p>ELIB_CV_OK : Normal</p> <p>ELIB_CV_NG : Abnormal</p>
Include file	srv_cv.h		
Functional description	<p>Broadcasts the inter-application transmittance event notification to applications. The event broadcasted by the interface is notified to the application that monitors the events corresponding to "Starting inter-application transmittance event monitoring".</p> <p>Use this function to exchange any events related to the CS service between applications transparently.</p> <p><Events to be notified></p> <p><u>Inter-application transmittance event notification:</u></p> <ul style="list-style-type: none"> - VoiceNotify_TransEvent <p>Structure of the inter-application transmittance event notification</p> <pre>typedef struct { int category; -> VoiceNotify</pre>		

	<pre> int subtype; -> Set the value of the argument type. int info; -> Set the value of the argument info. int subinfo; -> Set the value of the argument subinfo. union { unsigned char data[ELIB_CV_TRAN_DATA_SIZE]; -> Set the value of the argument data. } data ; } _ELIB_CV_EVENT ; #define ELIB_CV_TRAN_DATA_SIZE 256 /* Data size of the inter-application transmittance event */ </pre>
Related message	None
Necessary procedure	None
Note	<ul style="list-style-type: none"> - Up to eight notification events determined arbitrarily between applications can be notified. - How to identify notification events needs to be determined arbitrarily between applications. - The CS service has no concern in the data to be notified as an event, because the any value set by the application is notified.
Prohibition	The events that are not related the CS service is prohibited from being sent and received.
Use example	<pre> #define MUI_CV_EVT_VOICE_MSG VoiceNotify_TranEvent_01 /* Notification of results for an inquiry about the number of stored voice messages */ #define MUI_CV_CAU_INQ_SV_OK 0 /* Service inquiry OK */ /* Reject information structure */ typedef struct tagMUI_CV_USSD_REJECT_INF { unsigned char RestInf; /* Transmission restriction information */ } _MUI_CV_USSD_REJECT_INF ; /* Function to check the number of MUI stored voice messages */ void Mui_Voice_Msg_Cnt_Chk(_ELIB_CV_EVENT event ; /* USSD response result */) { _MUI_CV_USSD_REJECT_INF reject_inf ; /* Transmission restriction information */ int result ; /* Analysis result */ int ret ; /* Processing result */ - - /* Analyzing USSD response results */ Mui_USSD_Ind_Analysis (event->data.ussd_ind.str, &result, &reject_inf) ; - </pre>

	<pre>- /* Notifying of the inquiry results about the number of stored voice messages to applications */ ret = Elib_CV_Tran_Event_Broadcast (MUI_CV_EVT_VOICE_MSG, /* Inquiry result notification */ result, /* Analysis result */ 0, /* N/A */ (void *)&reject_inf); / Transmission restriction information */ - -</pre>
--	--

2.65 Processing of call time registration

2-65 Processing of call time registration			
Classification	Other services		
Function	Processing of call time registration		
Symbol	Elib_CV_TalkTimeSet		
Syntax	int Elib_CV_TalkTimeSet (time, kind);		
Argument	Type	I/O	Description
time	long	I	Call time (unit: seconds)
kind	int	I	Call type ELIB_CV_TTLTK_VOICE Voice communication ELIB_CV_TTLTK_PACKET Packet communication
Return value	Type	I/O	Description
Sts	int	O	Processing result ELIB_CV_OK : Normal ELIB_CV_NG : Abnormal
Include file	srv_cv.h		
Functional description	Registers call time and totalized call time.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example			

2.66 Processing of obtaining calling operation start time

6-66 Processing of obtaining calling operation start time			
Classification	Other services		
Function	Processing of obtaining calling operation start time		
Symbol	Elib_CV_Get_Calling_Start_Time		
Syntax	int Elib_CV_Get_Calling_Start_Time (void) ;		
Argument	Type	I/O	Description
None	void	-	-
Return value	Type	I/O	Description
ret	int	O	Calling operation start time settings 0 to 99 (seconds)
Include file	srv_cv.h		
Functional description	Obtains the settings of calling operation start time.		
Related message			
Necessary procedure			
Note			
Prohibition			
Use example			

2.67 Processing of setting of calling operation starttime

2-67 Processing of setting of calling operation start time			
Classification	Other services		
Function	Processing of setting of calling operation start time		
Symbol	Elib_CV_Set_Calling_Start_Time		
Syntax	int Elib_CV_Set_Calling_Start_Time (timer);		
Argument	Type	I/O	Description
timer	int	I	Calling operation start time settings 0 to 99 (seconds)
Return value	Type	I/O	Description
ret	int	O	ELIB_CV_OK : Normal end ELIB_CV_NG : Parameter error
Include file	srv_cv.h		
Functional description	Sets the calling operation start time.		
Related message			
Necessary procedure			
Note			
Prohibition			
Use example			

2.68 Obtaining absence incoming call history display setting

2-68 Obtaining absence incoming call history display setting			
Classification	Other services		
Function	Obtaining absence incoming call history display setting		
Symbol	Elib_CV_Get_Called_Record_ind		
Syntax	int Elib_CV_Get_Called_Record_ind (void);		
Argument	Type	I/O	Description
-	void	-	-
Return value	Type	I/O	Description
ret	int	O	ELIB_CV_ON : Display setting ON ELIB_CV_OFF: Display setting OFF
Include file	srv_cv.h		
Functional description	Obtains the display setting of the absence incoming call history.		
Related message			
Necessary procedure			
Note			
Prohibition			
Use example			

2.69 Processing of absence incoming call history display setting

2-69 Processing of absence incoming call history display setting			
Classification	Other services		
Function	Processing of absence incoming call history display setting		
Symbol	Elib_CV_Set_Called_Record_ind		
Syntax	int Elib_CV_Set_Called_Record_ind (mode);		
Argument	Type	I/O	Description
mode	Int	I	ELIB_CV_ON : Display setting ON ELIB_CV_OFF: Display setting OFF
Return value	Type	I/O	Description
ret	Int	O	ELIB_CV_OK : Normal end ELIB_CV_NG : Parameter error
Include file	srv_cv.h		
Functional description	Sets the display of the absence incoming call history.		
Related message			
Necessary procedure			
Note			
Prohibition			
Use example			

3. SMS Service

3.1 SMS Communication Functions Details

3-1 Start SMS communication monitoring			
Classification	SMS service/SMS communication		
Function	Start SMS communication monitoring		
Symbol	Elib_SMS_Request		
Syntax	int Elib_SMS_Request (unsigned int Ap_ID, int event, MsbFunc callback_func);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
event	int	I	Event to be reported * See Functional description below.
callback_func	MsbFunc	I	Callback function in which the event is reported
Return value	Type	I/O	Description
Ret	int	O	Processing result ELIB_SMS_OK : Normal end ELIB_SMS_NG : Abnormal end ELIB_SMS_PARAM_NG: Parameter error
Include file	srv_sms.h		
Functional description	<p>Registers the event to be reported to the AP side at a time when an event occurrence message from SMS-TAF was received.</p> <p>Multiple events can be specified in various combinations with the OR value. A parameter error occurs when an event type that cannot be specified was set. The specifiable event types are described below.</p> <pre> SMSNotify_MO_DATA_IND /* SMS-MO send indication notification */ SMSNotify_MT_DATA_IND /* SMS-MT receive indication notification */ SMSNotify_STORE_IND /* SMS message-stored notification SMSNotify_MEMORY_AVAILABLE_IND SMSNotify_ERROR_IND /* SMS error notification SMSNotify_REPORT_IND /* SMS delivery report received indication notification */ SMSNotify_CNMA_IND /* Receive response accepted notification event */ SMSNotify_ALL /* Specifies all SMS-related events </pre> <p>*1 SMSNotify_ALL indicates that the all specifiable event types are ORed.</p> <pre> SMSNotify_ALL = (SMSNotify_MO_DATA_IND SMSNotify_MT_DATA_IND SMSNotify_STORE_IND SMSNotify_MEMORY_AVAILABLE_IND SMSNotify_ERROR_IND SMSNotify_REPORT_IND </pre>		

	SMSNotify_CNMA_IND)
	For details on each event, see " 4. Event Structures ".
Related message	Messages from SMS-TAF
Necessary procedure	None
Note	None
Prohibition	None
Use example	None

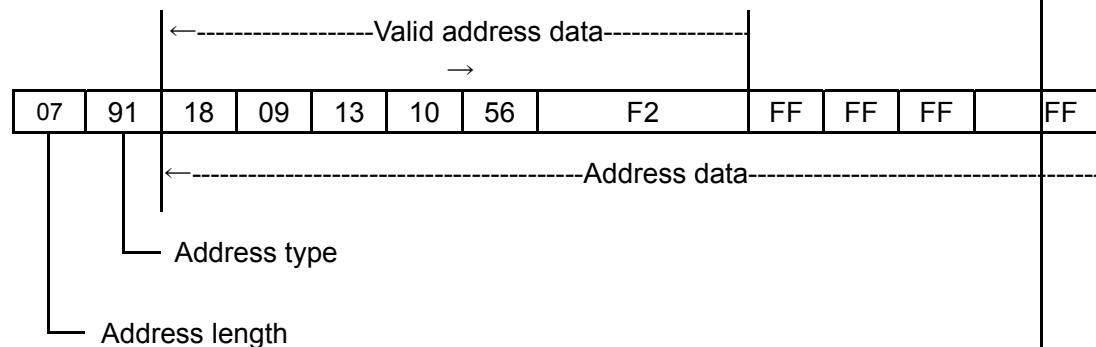
3.2 stop SMS communication monitoring

3-2 Stop SMS communication monitoring			
Classification	SMS service/SMS communication		
Function	Stop SMS communication monitoring		
Symbol	Elib_SMS_Cancel		
Syntax	Int Elib_SMS_Cancel (nsigned int Ap_ID, int event) ;		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Event	Int	I	Event to be canceled
Return value	Type	I/O	Description
Ret	Int	O	Processing result ELIB_SMS_OK : Normal end ELIB_SMS_NG : Abnormal end ELIB_SMS_PARAM_NG: Parameter error
Include file	srv_sms.h		
Functional description	Cancels the event to be reported to the AP side that has been registered by A-1 requesting SMS service event occurrence notification. Multiple events can be specified in various combinations with the OR value. For specifying events, see the events described in A-1 requesting SMS service event occurrence notification.		
Related message	None		
Necessary procedure	The events to be reported must have been requested by A-1 requesting SMS service event occurrence notification.		
Note	None		
Prohibition	None		
Use example	None		

3.3 stop SMS communication monitoring

3-3 SMS-MO send request			
Classification	SMS service/SMS communication		
Function	SMS-MO send request		
Symbol	Elib_SMS_SendData		
Syntax	int Elib_SMS_SendData (unsigned int Ap_ID, _ELIB_SMS_SND_DATA *SndData) ;		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
SndData	_ELIB_SMS_SND_DATA *	I	Pointer of the send request structure * See Functional description below.
Return value	Type	I/O	Description
Ret	Int	O	Processing result ELIB_SMS_OK : Normal end ELIB_SMS_NG : Abnormal end ELIB_SMS_PARAM_NG : Parameter error ELIB_SMS_SCA_NOT_SET : No service center address * Has not been set to the USIM card.
Include file	Srv_sms.h		
Functional description	<p>Makes a send request.</p> <ul style="list-style-type: none"> - Requests AT-BIN(+CMGS) to SMS-TAF. <p>/* Details on the second parameter, send request structure SndData */</p> <pre>typedef struct tagELIB_SMS_SND_DATA { unsigned char cr; /* Call ID number */ (Set 0.) _ELIB_SMS_RP_ADDRESS sca; /* Destination service center address */ See A-3-(1). /* (Set all zeros.) */ _ELIB_SMS_SUBMIT tpdu; /* TPDU data (SMS-SUBMIT) */ } _ELIB_SMS_SND_DATA;</pre>		
	<p>/* _ELIB_SMS_RP_ADDRESS (destination service center address) structure */</p> <pre>typedef struct tagELIB_SMS_RP_ADDRESS { unsigned charlen; /* Address length */ (0 to 11) Byte count of address type + byte count of valid address data unsigned char type; /* Address type */ unsigned char address[ELIB_SMS_RP_ADDRESS_MAX]; /* Address data[10] */ If the address type is: An alphanumeric character, GSM7bit code is used. Other than alphanumeric character, BCD code is used. After the valid address, set 1111 for each 4 bits. } _ELIB_SMS_RP_ADDRESS;</pre> <p><Memory image of destination service center address></p>		

Address type: 0x91 (International number, ISDN)
 Address data: 81903101652
 The memory image of the address above is shown below.



```

/* A-3-(2) SMS-MO send request TPDU data (SMS-SUBMIT) structure */
typedef struct tagELIB_SMS_SUBMIT {
    unsigned char    tp_msgInf;                /* Message information */
    unsigned char    tp_mr;                    /* Message reference information */
    /* Value used for message discrimination:
       Value added 1 to the Last-Used-TP-MR value of EFsmss */
    _ELIB_SMS_TP_DA  tp_da;                    /* Destination address information */
    /* See A-3-(3).
    unsigned char    tp_pid;                    /* Protocol identifier */
    unsigned char    tp_dcs;                    /* Data code system */
    unsigned char    tp_vp[ELIB_SMS_TPVP_MAX];
    /* Send cycle information[7] */
    unsigned char    tp_udl;                    /* User data length */
    /* If the data code system is:
       GSM7bit, the data is represented in 7-bit units.
       UCS2, the data is represented in 8-bit units.
    unsigned char    tp_ud[ELIB_SMS_SUBMIT_MAX];
    /* User data[140] */
    /* Short message character string
       (The format depends on the data code system.)
    } _ELIB_SMS_SUBMIT;
  
```

```

/* A-3-(3) _ELIB_SMS_TP_DA (destination address information) structure */
typedef struct tagELIB_SMS_TP_DA {
    unsigned char    len;                      /* Address length */ (0 to 20)
    /* Valid address data length in 4-bit units
    unsigned char    type;                      /* Address type */
    unsigned char    address[ELIB_SMS_TP_DA_MAX]; /* Address data[10] */
    /* If the address type is:
  
```

Alphanumeric character, GSM7bit code is used.

Other than alphanumeric character, BCD code is used.

If the address length is an odd number, set 1111 for the last 4 bits.

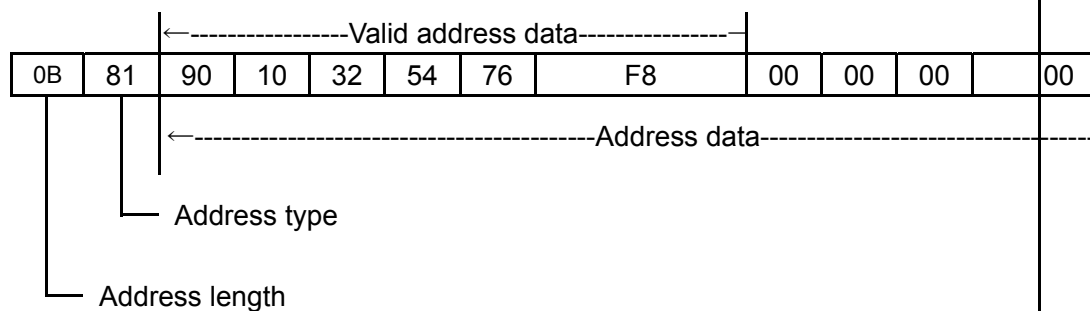
} _ELIB_SMS_TP_DA;

<Memory image of destination address information>

Address type: 0x81 (Unknown, ISDN)

Address data: 09012345678

The memory image of the address above is shown below.



/* A-3-(4) Details on address type */

7	6	5	4	3	2	1	0	
1	Type-of-number			Numbering-plan-identification				byte 1

Table A-3-(4)-1 Setting values of Type-of-number

Bit 7	bit 6	bit 5	bit 4	bit 3	Bit 2	bit 1	bit 0	Meaning
1	0	0	0					Unknown
1	0	0	1					International number
1	0	1	0					National number
1	0	1	1					Network number
1	1	0	0					Subscriber number
1	1	0	1					Alphanumeric character * Address data in GSM7bit code
1	1	1	0					Abbreviated number
1	1	1	1					Reserved for extension

Table A-3-(4)-2 Setting values of Numbering-plan-identification

Bit 7	bit 6	bit 5	bit 4	bit 3	Bit 2	bit 1	bit 0	Meaning
1				0	0	0	0	Unknown
1				0	0	0	1	ISDN
1				0	0	1	1	X.121
1				0	1	0	0	Telex

1				1	0	0	0	National
1				1	0	0	1	Personal
1				1	0	1	0	ERMES
1				1	1	1	1	Reserved for extension

/* A-3-(5) Message information (SMS-SUBMIT) */

7	6	5	4	3	2	1	0	
TP-RP	TP-UDHI	TP-SRR	TP-VPF	TP-RD	TP-MTI			Byte 1

Table A-3-(5) Setting values of message information (SMS-SUBMIT)

Name	Meaning	Size (bit)	Note
TP-MTI	Message type information	2	bit1 bit0 0 1 SMS-SUBMIT(MS→SC)
TP-RD	Duplicate message exclusion information	1	0: Receives duplicate message (for normal send) 1: Rejects duplicate message (for resend)
TP-VPF	Send cycle format	2	bit4 bit3 0 0 TP-VP field does not exist. 1 0 Relative format (1-byte fixed) 0 1 Enhanced format (7-byte fixed) 1 1 Absolute format (7-byte fixed)
TP-SRR	Status report request	1	0: Status report not-required 1: Status report required
TP-UDHI	User data header information	1	0: UD includes a short message only. 1: UD includes the additional header information.
TP-RP	Return path information	1	0: The return path is not set. 1: The return path is set.

Related message

When the sending succeeded, the following event is reported.

SMSNotify_MO_DATA_IND

/* SMS-MO send indication */

When the sending failed, the following error event is reported.

SMSNotify_ERROR_IND /* Error indication */

Necessary procedure

None

Note

The function does not perform the conformability check of the TPDU data.

Since the function does not perform the code conversion, previously perform the code conversion at the calling side by using the character conversion library of the data switch service (the Cnvl_ChkExcCode function and Cnvl_StrCodeCnv function).

Prohibition	None
Use example	None

3.4 SMS-MT receive response

3-4 SMS-MT receive response			
Classification	SMS service/SMS communication		
Function	SMS-MT receive response		
Symbol	Elib_SMS_RcvRes		
Syntax	int Elib_SMS_RcvRes (unsigned int Ap_ID, _ELIB_SMS_RCV_RES *RcvRes) ;		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
RcvRes	_ELIB_SMS_RCV_RES *	I	Pointer of the receive response structure * See Functional description below.
Return value	Type	I/O	Description
Ret	Int	O	Processing result ELIB_SMS_OK : Normal end ELIB_SMS_NG : Abnormal end ELIB_SMS_PARAM_NG : Parameter error
Include file	srv_sms.h		
Functional description	<p>Reports that the data was normally received at the SMS-MT receive indication event, delivery report received notification event, or message-stored notification event to the MPS side.</p> <ul style="list-style-type: none"> - Requests AT-BIN(+CNMA) to SMS-TAF. <pre> /* Details on the second parameter, receive response structure RcvRes */ typedef struct tagELIB_SMS_RCV_RES { unsigned char cr; /* Call ID number */ Set the call ID number of the SMS-MT receive indication event, delivery report received notification event, or message-stored notification event. _ELIB_SMS_DELIVER_ACK tpdu; /* TPDU data (SMS-DELIVER-REPORT for RP-ACK) */ See A-4-(1). } _ELIB_SMS_RCV_RES; /* A-4-(1) Receive response TPDU data (SMS-DELIVER-REPORT for RP-ACK) structure */ typedef struct tagELIB_SMS_DELIVER_ACK { unsigned char tp_msgInf; /* Message information */ See A-4-(2). unsigned char tp_pi; /* Presence or absence of option parameters */ See 4.4.4. </pre>		

	unsigned char	tp_pid;	/* Protocol identifier */ See 4.4.1.																		
	identifier bit of tp_pi is 1.		* Valid only when the protocol																		
	unsigned char	tp_dcs;	/* Data code system */ See 4.4.2.																		
	system bit of tp_pi is 1.		* Valid only when the data code																		
	unsigned char	tp_udl;	/* User data length */																		
	bit of tp_pi is 1.		* Valid only when the user data length																		
			If the data code system is:																		
	7-bit units.		GSM7bit, the data is represented in																		
			UCS2, the data is represented in																		
	8-bit units.																				
	unsigned char	tp_ud[ELIB_SMS_DELIVER_ACK_MAX];	/*																		
	User data[159]	*/ * See Note.																			
			Short																		
	message character string		(The format																		
			depends on the data code system.)																		
	} _ELIB_SMS_DELIVER_ACK;																				
	/* A-4-(2) Message information (SMS-DELIVER-REPORT) */																				
	7	6	5	4	3	2	1	0													
	0	TP-UDHI	0	0	0	0	TP-MTI		byte 1												
	Table A-4-(2) Setting values of message information (SMS-DELIVER-REPORT)																				
	<table><tr><th>Name</th><th>Meaning</th><th>Size (bit)</th><th>Note</th></tr><tr><td>TP-MTI</td><td>Message information type</td><td>2</td><td>bit1 bit0 0 0 SMS-DELIVER REPORT(MS → SC)</td></tr><tr><td>TP-UDHI</td><td>User data header information</td><td>1</td><td>0: UD includes a short message only. 1: UD includes the additional header information.</td></tr></table>									Name	Meaning	Size (bit)	Note	TP-MTI	Message information type	2	bit1 bit0 0 0 SMS-DELIVER REPORT(MS → SC)	TP-UDHI	User data header information	1	0: UD includes a short message only. 1: UD includes the additional header information.
Name	Meaning	Size (bit)	Note																		
TP-MTI	Message information type	2	bit1 bit0 0 0 SMS-DELIVER REPORT(MS → SC)																		
TP-UDHI	User data header information	1	0: UD includes a short message only. 1: UD includes the additional header information.																		
Related message	None																				
Necessary procedure	None																				
Note	The function does not perform the conformability check of the TPDU data. Since the function does not perform the code conversion, previously perform the code conversion at the calling side by using the character conversion library of the data switch service (the Cnvl StrCodeCnv function).																				

Prohibition	None
Use example	None

3.5 SMS-MT receive failure response

3-5 SMS-MT receive failure response			
Classification	SMS service/SMS communication		
Function	SMS-MT receive failure response		
Symbol	Elib_SMS_RcvNGRes		
Syntax	int Elib_SMS_RcvNGRes (unsigned int Ap_ID, _ELIB_SMS_RCVNG_RES *RcvNgRes);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
RcvNgRes	_ELIB_SMS_RCVNG_RES *	I	Pointer of the receive failure response structure * See Functional description below.
Return value	Type	I/O	Description
Ret	Int	O	Processing result ELIB_SMS_OK : Normal end ELIB_SMS_NG : Abnormal end ELIB_SMS_PARAM_NG : Parameter error
Include file	srv_sms.h		
Functional description	<p>Reports that the data reception failed at the SMS-MT receive indication event or delivery report received notification event to the MPS side.</p> <ul style="list-style-type: none"> - Requests AT-BIN(+CNMA) to SMS-TAF. <pre> /* Details on the second parameter, receive failure response structure RcvNgRes */ typedef struct tagELIB_SMS_RCVNG_RES { unsigned char cr; /* Call ID number */ Set the call ID number of the SMS-MT receive indication event or delivery report received notification event. _ELIB_SMS_DELIVER_ERROR tpdu; /* TPDU data (SMS-DELIVER-REPORT for RP-ERROR)*/ See A-5-(1). } _ELIB_SMS_RCVNG_RES; /* A-5-(1) Receive failure response TPDU data (SMS-DELIVER-REPORT for RP-ERROR) structure */ typedef struct tagELIB_SMS_DELIVER_ERROR { unsigned char tp_msgInf; /* Message information */ See A-4-(2). unsigned char tp_fcs; /* Error cause */ unsigned char tp_pi; /* Presence or absence of option parameters */ unsigned char tp_pid; /* Protocol identifier */ /* Valid only when the protocol identifier bit of tp_pi is 1. */ unsigned char tp_dcs; /* Data code system */ </pre>		

	<pre> /* Valid only when the data code system bit of tp_pi is 1. */ unsigned char tp_udl; /* User data length */ /* Valid only when the user data length bit of tp_pi is 1. */ If the data code system is: GSM7bit, the data is represented in 7-bit units. UCS2, the data is represented in 8-bit units. unsigned char tp_ud[ELIB_SMS_DELIVER_ERROR_MAX]; /*User data[158] *//*See Note Short message character string (The format depends on the data code system.) } _ELIB_SMS_DELIVER_ERROR; </pre>
Related message	None
Necessary procedure	None
Note	<p>The function does not perform the conformability check of the TPDU data.</p> <p>Since the function does not perform the code conversion, previously perform the code conversion at the calling side by using the character conversion library of the data switch service (the Cnvl_StrCodeCnv function).</p>
Prohibition	None
Use example	None

3.6 SMS receive memory check

3-6 SMS receive memory check			
Classification	SMS service/SMS communication		
Function	SMS receive memory check		
Symbol	Elib_SMS_RcvMemChk		
Syntax	int Elib_SMS_RcvMemChk (unsigned int Ap_ID, _ELIB_SMS_RCV_MEM_CHK *RcvMemChk) ;		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
RcvMemChk	_ELIB_SMS_RCV_MEM_CHK *	I	Pointer of the receive memory check structure * See Functional description below.
Return value	Type	I/O	Description
Ret	Int	O	Processing result ELIB_SMS_OK : Normal end ELIB_SMS_NG : Abnormal end ELIB_SMS_PARAM_NG : Parameter error
Include file	srv_sms.h		
Functional description	<p>Sends the information that the SMS-MT reception is enabled to the MPS side.</p> <ul style="list-style-type: none"> - Requests AT-BIN(+CMGS) to SMS-TAF. <pre>/* Details on the second parameter, receive memory check structure RcvMemChk */ typedef struct tagELIB_SMS_RCV_MEM_CHK { unsigned char cr; /* Call ID number */ (Set 0.) } _ELIB_SMS_RCV_MEM_CHK;</pre>		
Related message	<p>When the sending succeeded, the following event is reported.</p> <pre>SMSNotify_MEMORY_AVAILABLE_IND /* Receive memory check response */</pre> <p>When the sending failed, the following error event is reported.</p> <pre>SMSNotify_ERROR_IND /* Error indication */</pre>		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example	None		

3.7 stop SMS communication monitoring

3-7 Abort SMS communication processing			
Classification	SMS service/SMS communication		
Function	Abort SMS communication processing		
Symbol	Elib_SMS_Abort_Req		
Syntax	int Elib_SMS_Abort_Req (unsigned int Ap_ID, _ELIB_SMS_ABORT *Abort) ;		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Abort	_ELIB_SMS_ABORT *	I	Pointer of the abort request structure * See Functional description below.
Return value	Type	I/O	Description
Ret	int	O	Processing result ELIB_SMS_OK : Normal end ELIB_SMS_NG : Abnormal end ELIB_SMS_PARAM_NG : Parameter error
Include file	srv_sms.h		
Functional description	Requests to abort the communication processing. <pre>/* Details on the second parameter, abort request structure Abort */ typedef struct tagELIB_SMS_ABORT { unsigned char cr; /* Call ID number */ (Set 0.) } _ELIB_SMS_ABORT;</pre>		
Related message	None		
Necessary procedure	None		
Note	Use the function after requesting the receive memory check (A-6 SMS receive memory check).		
Prohibition	None		
Use example	None		

3.8 Get SMS communication status

3-8 Get SMS communication status			
Classification	SMS service/SMS communication		
Function	Get SMS communication status		
Symbol	Elib_SMS_Get_Com_Status		
Syntax	int Elib_SMS_Get_Com_Status (unsigned int Ap_ID) ;		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Return value	Type	I/O	Description
Sts	int	O	SMS communication status ELIB_SMS_COM_STATUS_EMPTY : Communication terminated (idle) ELIB_SMS_COM_STATUS_ACTIVE : Active ELIB_SMS_COM_STATUS_SND : Sending ELIB_SMS_COM_STATUS_RCV : Receiving ELIB_SMS_NG : Abnormal end ELIB_SMS_PARAM_NG : Parameter error
Include file	srv_sms.h		
Functional description	Acquires the current SMS communication status.		
Related message	None		
Necessary procedure	None		
Note	None		
Prohibition	None		
Use example	None		

4. Equipment Service

4.1 Equipment Service event notice request start

4-1 Equipment Service event notice request start			
Classification	Equipment Service		
Function	Equipment Service event notice request start		
Symbol	Elib_WDC_Request		
Syntax	int Elib_WDC_Request(ap_id, event, func);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
event	int	I	Type of an event to be caused. See "Functional description."
Func	MsbFunc	I	Pointer to a function to be called back at event occurrence. See "Functional description."
Return value	Type	I/O	Description
Sts	int	O	Processing result ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_PARAERR : Parameter error
Include file	srv_wdc.h msb/msb.h		
Functional description	<p>- In Equipment Service, any application that needs the following events must start an event notice request by calling this interface (function).</p> <p>- Event list (values specified with the argument Event)</p> <pre>WDCNotify_ALL /* All Equipment Service events */ WDCNotify_BattLv /* Battery level change event */ WDCNotify_EarPhone /* Earphone connection status change event */ WDCNotify_OpenClose /* Open/closed status change event */ WDCNotify_Charge /* Charger status change event */ WDCNotify_NvmAlarm /* Nonvolatile alarm status change event */ WDCNotify_ActMode /* Operation mode status change event */ WDCNotify_Valclrind /* Setup clear request event */ WDCNotify_Prohibit /* Banned-operation setup event */ WDCNotify_ConInfo /* UIM connection information notice event */ WDCNotify_UnCorr /* Wrong card insertion event */ WDCNotify_Monitor /* Monitor mode response event */ WDCNotify_UsbInfj /* USB connection status change event */ WDCNotify_IrInfj /* Ir connection status change event */ WDCNotify_SwupErrDsp /* SWUP abnormal indication notice event */ WDCNotify_Secret /* Secret status change notice event */</pre> <p>* Events will be added as requested by applications.</p> <p>- Listed below is the format of the function called back when the event occurs.</p> <pre>void func(MsbObject *client, MsbObject *server, void *sendData, void *recvData, void *data, MsbEnvironment *ev);</pre>		

	* The MSB is used to notify of events.
Related message	-
Necessary procedure	- Before executing the function, each APL must generate an MSB object, using msb_Object_Create.
Note	<ul style="list-style-type: none"> - WDCNotify_Valclrind event Notifies of this event when requesting a setup reset. If an APL must reset data held in it, it should receive this event and perform setup reset for the data. (As for setup data provided by the telephone function interface, the telephone function interface resets the data.) - Equipment Service does not draw the icons on the upper row shown below. <ol style="list-style-type: none"> 1) Battery icon 2) Banned-operation icon 3) USB icon 4) Ir connection icon 5) Secret icon
Prohibition	None
Use example	<pre> /* Function called back when the event is acquired */ void func(MsbObject *client, MsbObject *server, void *sendData, void *recvData, void *data, MsbEnvironment *ev); unsigned int ap_id; /* Application ID */ int event; /* Type of an event to be caused */ int ret; /* Function's return value */ /* Parameter setup for registration */ ap_id = AP_ID_AP02; event = WDCNotify_BattLv; /* Battery level change event specified */ /* Function call */ ret = Elib_WDC_Request(ap_id, event, (MsbFunc)(func)); </pre>

4.2 Equipment Service event notice request stop

4-2 Equipment Service event notice request stop			
Classification	Equipment Service		
Function	Equipment Service event notice request stop		
Symbol	Elib_WDC_Cancel		
Syntax	int Elib_WDC_Cancel(ap_id, event);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Event	int	I	Type of an event whose notification is to be released * See " Equipment Service event notice request start " for explanations about event types.
Return value	Type	I/O	Description
Sts	int	O	Processing result ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_NODATA : Not registered ELIB_WDC_PARAERR : Parameter error
Include file	srv_wdc.h		
Functional description	This function stops notifying of the specified Equipment Service event.		
Related message	-		
Necessary procedure	- Before executing this function, each APL must generate an MSB object, using msb_Object_Create.		
Note			
Prohibition	None		
Use example	<pre> unsigned int ap_id; /* Application ID */ int event; /* Type of an event to be stopped */ int ret; /* Function's return value */ /* Parameter setup for registration */ ap_id = AP_ID_AP02; event = WDCNotify_BattLv; /* Specifies a battery level change event. */ /* Function call */ ret = Elib_WDC_Cancel(ap_id, event); </pre>		

4.3 Earphone mode setup

4-3 Earphone mode setup			
Classification	Equipment Service		
Function	Earphone mode setup		
Symbol	Elib_WDC_Set_EarphoneMode		
Syntax	int Elib_WDC_Set_EarphoneMode(mode);		
Argument	Type	I/O	Description
Mode	Int	I	ELIB_WDC_EAR : Earphone only ELIB_WDC_SOUND : Earphone + speaker
Return value	Type	I/O	Description
Sts	Int	O	Processing result ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_PARAERR : Parameter error
Include file	srv_wdc.h		
Functional description	The earphone mode is switched. - The default value is "earphone + speaker (ELIB_WDC_SOUND)."		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int mode; /* Variable for specifying the earphone mode */ int ret; /* Function's return value */ /* Parameter setup */ mode = ELIB_WDC_EAR; /* Function call */ ret = Elib_WDC_Set_EarphoneMode(mode);</pre>		

4.4 Manner mode ON/OFF

4-4 Manner mode ON/OFF			
Classification	Equipment Service		
Function	Manner mode ON/OFF		
Symbol	Elib_WDC_OnOff_Safety		
Syntax	int Elib_WDC_OnOff_Safety(mode);		
Argument	Type	I/O	Description
Mode	Int	I	ELIB_WDC_SAFETY_ON : The manner mode is ON. ELIB_WDC_SAFETY_OFF : The manner mode is OFF
Return value	Type	I/O	Description
Sts	Int	O	Processing result ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_PARAERR : Parameter error
Include file	srv_wdc.h		
Functional description	<p>The function specifies whether to turn ON or OFF the manner mode.</p> <p>* One of the following processing types is performed depending on the current audio communication status.</p> <ol style="list-style-type: none"> 1. Waiting Tone is suppressed. 2. Conversation or three-party conversation in progress, ringing, or call being on hold The mike amp gain is changed. The manner mode start confirmatory sound is generated. 3. Call incoming Ringing tone is suppressed. <p>* Processing types not mentioned above, such as screen control and operator's panel processing, should be implemented by applications.</p> <p>- The default value is "Manner mode OFF (ELIB_WDC_SAFETY_OFF)."</p>		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int mode; /* Variable for specifying whether to turn ON or OFF the manner mode */ int ret; /* Function's return value */ /** Parameter setup */ mode = ELIB_WDC_SAFETY_ON;</pre>		

	<pre>/** Function call */ ret = Elib_WDC_OnOff_Safety(mode);</pre>
--	--

4.5 Manner mode setup

4-5 Manner mode setup			
Classification	Equipment Service		
Function	Manner mode setup		
Symbol	Elib_WDC_Set_SafetyMode		
Syntax	int Elib_WDC_Set_SafetyMode(mode);		
Argument	Type	I/O	Description
Mode	Int	I	ELIB_WDC_SAFETY_SILENT: Expansion by PF vendor ELIB_WDC_SAFETY_VIB : Manner mode ELIB_WDC_SAFETY_ORG : Original manner mode
Return value	Type	I/O	Description
Ret	int	O	Processing result ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_PARAERR : Parameter error
Include file	srv_wdc.h		
Functional description	The manner mode is set up. - The default value is "manner mode (ELIB_WDC_SAFETY_VIB)."		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int mode; /* Variable for specifying the manner mode */ int ret; /* Function's return value */ /** Parameter setup */ mode = ELIB_WDC_SAFETY_SILENT; /** Function call */ ret = Elib_WDC_Set_SafetyMode(mode);</pre>		

4.6 Original manner mode serup

4-6 Original manner mode setup			
Classification	Equipment Service		
Function	Original manner mode setup		
Symbol	Elib_WDC_Set_OrgSafetyMode		
Syntax	int Elib_WDC_Set_OrgSafetyMode(mode);		
Argument	Type	I/O	Description
Mode	_ELIB_WDC_OrgSafetyMode *	I	See "Functional description."
Return value	Type	I/O	Description
Ret	Int	O	ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_PARAERR : Parameter error
Include file	srv_wdc.h		
Functional description	<p>- The original manner mode is set up.</p> <p><_ELIB_WDC_OrgSafetyMode structure description></p> <pre> typedef struct tag_EWDC_OrgSafetyMode { _ELIB_WDC_OrgSafetyVolume volume /* Structure for specifying sound volume for the original manner mode */ unsigned char memo; /* Voice message memo ON/OFF ON : ELIB_WDC_MEMO_ON *1 OFF: ELIB_WDC_MEMO_OFF */ unsigned char vibrator; /* Vibrator ON/OFF ON: ELIB_WDC_VIB_ON OFF: ELIB_WDC_ unsigned char voice; /* Voice/memo confirmatory sound ON: ELIB_WDC_VMTONE_ON OFF: ELIB_WDC_VMTONE_OFF */ unsigned char button; /* Button confirmation ON/OFF ON: ELIB_WDC_BUTTON_ON OFF: ELIB_WDC_BUTTON_OFF */ unsigned char micgain; /* Mike sensitivity setup Normal: ELIB_WDC_MICGAIN_NORMAL UP : ELIB_WDC_MICGAIN_UP */ unsigned char alarm; /* Low-voltage alarm sound ON: ELIB_WDC_LOWBAT_ON OFF: ELIB_WDC_LOWBAT_OFF */ } _ELIB_WDC_OrgSafetyMode; </pre> <p><_ELIB_WDC_OrgSafetyVolume structure description></p> <p>Structure for specifying the sound volume of telephone ringing, e-mail ringing, and alarm tone separately</p> <pre> typedef struct tag_EWDC_OrgSafetyVolume { </pre>		

	<pre> unsigned char tel; /* Sound volume of telephone ringing */ unsigned char mail; /* Sound volume of e-mail ringing */ unsigned char mealarm; /* Sound volume of alarm tone */ } _ELIB_WDC_OrgSafetyVolume ; * The values that can be specified include 0 (silent), 1 to 6 (volume levels 1 to 6), and ELIB_WDC_STEPTONE (step tone). - Listed below are the default values for the original manner mode. Voice message memo : OFF Vibrator : ON Ringing : Silent for all types of ringing Memo confirmatory sound : ON Button confirmatory sound : OFF Mike sensitivity for conversation by phone: UP Low-voltage alarm : OFF *1: Described below are how the voice memo calling time in the original mode is obtained. <When M55 voice message memo is ON> Use the "Elib_REC_VoiceMemoStatus_Get" Efunction for ELIB record and playback. <When M55 voice message memo is OFF> The high-order application shall use the default value ELIB_WDC_DENTIME_DEFFALT(8). * This is because, the voice message memo calling time obtained by "Elib_REC_VoiceMemoStatus_Get" is undefined (not guaranteed) if M55 voice message memo is OFF. </pre>
Related message	None
Necessary procedure	
Note	
Prohibition	None
Use example	<pre> _ELIB_WDC_OrgSafetyMode mode; /* Structure for specifying the original manner mode */ int ret; /* Function's return value */ /** Structure member setup */ mode.volume.tel = 0; mode.volume.mail = 6; mode.volume.mealarm = ELIB_WDC_STEPTONE; mode.memo = ELIB_WDC_MEMO_ON; mode.vibrator = ELIB_WDC_VIB_ON; mode.voice = ELIB_WDC_VMTONEON; mode.button = ELIB_WDC_BUTTON_ON; mode.micgain = ELIB_WDC_MICGAIN_NORMAL; </pre>

```
mode.alarm = ELIB_WDC_LOWBAT_ON;

/** Parameter setup */

/** Function call */
ret = Elib_WDC_Set_OrgSafetyMode(&mode);
```

4.7 Vibrator operation start/stop

4-7 Vibrator operation start/stop			
Classification	Equipment Service		
Function	Vibrator operation start/stop		
Symbol	Elib_WDC_StartStop_Vib		
Syntax	int Elib_WDC_StartStop_Vib(mode, VibID, kind);		
Argument	Type	I/O	Description
Mode	int	I	ELIB_WDC_VIB_DIRECT: Vibration starts with a pattern specified with VibID. ELIB_WDC_VIB_STOP : Vibration stops. ELIB_WDC_VIB_DIRECT_FOR_JAVA: i-appli software-dependent forced vibration
VibID	int	I	Specify a vibration ID, which must be the one obtained with "Elib_WDC_Get_VibTitle0." This argument is valid only when the specified mode is ELIB_WDC_VIB_DIRECT or ELIB_WDC_VIB_DIRECT_FOR_JAVA. * It is invalid if the specified mode is ELIB_WDC_VIB_STOP.
Kind	int	I	* Unused
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_PARAERR : Parameter error
Include file	srv_wdc.h		
Functional description	<p>The vibrator starts or stops vibrating.</p> <ul style="list-style-type: none"> - If the argument "mode" specifies ELIB_WDC_VIB_DIRECT, the vibrator can start with the specified vibration ID. - If you want to force the vibrator to operate as directed by i-appli software from JAVA, call this function by setting the argument "mode" with <code>ELIB_WDC_VIB_DIRECT_FOR_JAVA</code> - Stop the vibrator when i-appli software becomes inactive if the vibrator has been forced to operate as directed by i-appli software from JAVA. <p>* This function is valid only for demonstrative vibration with M54 vibrator specified and for i-appli software-dependent forced vibration.</p>		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<Starting vibration by specifying a vibration ID>		

	<pre> int mode; /* Variable for specifying whether to start or stop vibrator operation */ int ret; /* Function's return value */ _ELIB_WDC_VIB_TITLE VibTitle; /* Structure for holding a vibration ID */ int vib_id; /* Variable for specifying a vibration ID */ mode = ELIB_WDC_VIB_DIRECT; ret = Elib_WDC_Get_VibTitle(1, &VibTitle); /* Vibration ID acquisition */ vib_id = VibTitle.VibID; ret = Elib_WDC_StartStop_Vib(mode, vib_id, 0); </pre>
--	--

4.8 Vibrator pattern setup

4-8 Vibration pattern setup			
Classification	Equipment Service		
Function	Vibration pattern setup		
Symbol	Elib_WDC_Set_VibMode		
Syntax	int Elib_WDC_Set_VibMode(VibID, kind);		
Argument	Type	I/O	Description
VibID	Int	I	<p>A vibration pattern is specified.</p> <p>ELIB_WDC_VIB_PAT0 : Pattern No.0 (OFF)</p> <p>ELIB_WDC_VIB_PAT1 : Pattern No.1 (pattern 1)</p> <p>ELIB_WDC_VIB_PAT2 : Pattern No.2 (pattern 2)</p> <p>ELIB_WDC_VIB_PAT3 : Pattern No.3 (pattern 3)</p> <p>ELIB_WDC_VIB_PATMELO: Pattern No.4 (linked with a melody)</p> <p>* The specified pattern number must be the one obtained with "Elib_WDC_Get_VibTitle."</p>
Kind	Int	I	<p>ELIB_WDC_VIB_VOICE : Audiophone ringing setup</p> <p>ELIB_WDC_VIB_MAIL : E-mail ringing setup</p> <p>ELIB_WDC_VIB_MSG_R : R-message ringing setup</p> <p>ELIB_WDC_VIB_MSG_F : F-message ringing setup</p> <p>ELIB_WDC_VIB_TV_VOICE : Videophone ringing setup</p> <p>* A parameter error is assumed if any other pattern is specified.</p>
Return value	Type	I/O	Description
Ret	Int	O	<p>Processing result</p> <p>ELIB_WDC_OK : Normal end</p> <p>ELIB_WDC_NG : Abnormal end</p> <p>ELIB_WDC_PARAERR : Parameter error</p>
Include file	srv_wdc.h		
Functional description	<ul style="list-style-type: none"> - A vibration pattern is specified for a vibrator. - Specifying an incoming call type with the argument "kind" causes the vibrator to operate with a pattern specified for the call type. <p>* The default value for each of the incoming audiophone call, e-mail, R-message, F-message, and videophone call is OFF.</p>		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre><Setup for specifying to link the e-mail ringing with a melody > int VibID; /* Variable for specifying a vibration pattern */ int kind; /* Variable for specifying an incoming call type */ int ret1,ret2; /* Function's return value */</pre>		

```
_ELIB_WDC_VIB_TITLE Vib_DATA;  
/* Structure for holding pattern name and vibration ID */  
  
/** Parameter setup **/  
ret1 = Elib_WDC_Get_VibTitle(4, &Vib_DATA);  
/* Vibration ID acquisition */  
  
VibID = Vib_DATA.VibID ;  
kind = ELIB_WDC_VIB_MAIL ; /* Selecting e-mail ringing */  
  
/** Function call **/  
ret2 = Elib_WDC_Set_VibMode(VibID, kind);
```

4.9 Monitor mode start/stop

4-9 Secret mode setup			
Classification	Equipment Service		
Function	Secret mode setup		
Symbol	Elib_WDC_Set_SecretMode		
Syntax	int Elib_WDC_Set_SecretMode(mode);		
Argument	Type	I/O	Description
Mode	int	I	ELIB_WDC_NORMAL_MODE : Normal mode ELIB_WDC_SECRET_MODE : Secret mode ELIB_WDC_SECRET_SMODE: Secret-only mode
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_PARAERR : Parameter error
Include file	srv_wdc.h		
Functional description	The secret mode is set up. - The telephone function interface does not perform key icon display control in the secret mode.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int mode; /* Variable for specifying the secret mode */ int ret; /* Function's return value */ /** Parameter setup */ mode = ELIB_WDC_SECRET_SMODE; ret = Elib_WDC_Set_SecretMode(mode);</pre>		

4.10 Secret data type setup

4-10 Secret data type setup			
Classification	Equipment Service		
Function	Secret data type setup		
Symbol	Elib_WDC_Set_SecretData		
Syntax	int Elib_WDC_Set_SecretData(kind);		
Argument	Type	I/O	Description
Kind	int	I	ELIB_WDC_SECRETDATA : Secret data ELIB_WDC_NORMALDATA : Ordinary data
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_PARAERR : Parameter error
Include file	srv_wdc.h		
Functional description	A display data type is specified for the secret mode. - The telephone function interface does not perform key icon display control in the secret mode.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int kind; /* Variable for specifying a data type */ int ret; /* Function's return value */ /** Parameter setup **/ kind = ELIB_WDC_SECRETDATA; ret = Elib_WDC_Set_SecretData(kind);</pre>		

4.11 Terminal lock (all-feature lock) ON/OFF

4-11 Terminal lock (all-feature lock) ON/OFF			
Classification	Equipment Service		
Function	Terminal lock (all-feature lock) ON/OFF		
Symbol	Elib_WDC_OnOff_TrmLock		
Syntax	int Elib_WDC_OnOff_TrmLock(mode);		
Argument	Type	I/O	Description
Mode	Int	I	ELIB_WDC_TRM_ON : Terminal lock (all-feature lock) ON ELIB_WDC_TRM_OFF : Terminal lock (all-feature lock) OFF
Return value	Type	I/O	Description
Ret	Int	O	ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_PARAERR : Parameter error
Include file	srv_wdc.h		
Functional description	The terminal lock (all-feature lock) is turned ON/OFF. - The telephone function interface does not perform banned-operation icon display control.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int mode; /* Variable for specifying whether to turn ON or OFF the terminal lock */ int ret; /* Function's return value */ /** Parameter setup */ mode = ELIB_WDC_TRM_ON; /** Function call */ ret = Elib_WDC_OnOff_TrmLock(mode);</pre>		

4.12 PIM loc ON/OFF

4-12 PIM lock ON/OFF			
Classification	Equipment Service		
Function	PIM lock ON/OFF		
Symbol	Elib_WDC_OnOff_PIMLock		
Syntax	int Elib_WDC_OnOff_PIMLock(mode);		
Argument	Type	I/O	Description
mode	Int	I	ELIB_WDC_PIM_ON : PIM lock ON ELIB_WDC_PIM_OFF : PIM lock OFF
Return value	Type	I/O	Description
Ret	Int	O	ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_PARAERR : Parameter error
Include file	srv_wdc.h		
Functional description	The PIM lock is turned ON/OFF. - The telephone function interface does not perform banned-operation icon display control.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre> int mode; /* Variable for specifying whether to turn ON or OFF the PIM lock */ int ret; /* Function's return value */ /** Parameter setup */ mode = ELIB_WDC_PIM_ON; /** Function call */ ret = Elib_WDC_OnOff_PIMLock(mode); </pre>		

4.13 Keypad-activated dialing ban ON/OFF

4-13 Keypad-activated dialing ban ON/OFF			
Classification	Equipment Service		
Function	Keypad-activated dialing ban ON/OFF		
Symbol	Elib_WDC_OnOff_DialSend		
Syntax	int Elib_WDC_OnOff_DialSend(mode);		
Argument	Type	I/O	Description
mode	Int	I	ELIB_WDC_DIAL_ON : Keypad-activated dialing ban ON ELIB_WDC_DIAL_OFF : Keypad-activated dialing ban OFF
Return value	Type	I/O	Description
ret	Int	O	ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_PARAERR : Parameter error
Include file	srv_wdc.h		
Functional description	Keypad-activated dialing ban is turned ON/OFF. - The telephone function interface does not perform banned-operation icon display control.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int mode; /* Variable for specifying whether to turn ON or OFF the keypad-activated dialing ban */ int ret; /* Function's return value */ /** Parameter setup */ mode = ELIB_WDC_DIAL_ON; /** Function call */ ret = Elib_WDC_OnOff_DialSend(mode);</pre>		

4.14 Charge start/end confirmatory sound setup

4-14 Charge start/end confirmatory sound setup			
Classification	Equipment Service		
Function	Charge start/end confirmatory sound setup		
Symbol	Elib_WDC_Set_ChargeSound		
Syntax	int Elib_WDC_Set_ChargeSound(mode);		
Argument	Type	I/O	Description
Mode	Int	I	ELIB_WDC_CHARGESND_ON: Confirmatory sound ON ELIB_WDC_CHARGESND_OFF: Confirmatory sound OFF
Return value	Type	I/O	Description
ret	Int	O	ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_PARAERR : Parameter error
Include file	srv_wdc.h		
Functional description	Whether to generate confirmatory sound at the start and end of battery charge is specified.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int mode; /* Variable for specifying whether to turn ON or OFF confirmatory sound */ int ret; /* Function's return value */ /** Parameter setup */ mode = ELIB_WDC_CHARGESND_ON; /** Function call */ ret = Elib_WDC_Set_ChargeSound(mode);</pre>		

4.15 Rear LCD sleep mode ON/OFF

4-15 Rear LCD sleep mode ON/OFF			
Classification	Equipment Service		
Function	Rear LCD sleep mode ON/OFF		
Symbol	Elib_WDC_OnOff_BackLcdSleep		
Syntax	Int Elib_WDC_OnOff_BackLcdSleep (mode);		
Argument	Type	I/O	Description
mode	Int	I	ELIB_WDC_ON: Sleep mode ON ELIB_WDC_OFF: Sleep mode OFF
Return value	Type	I/O	Description
Ret	Int	O	ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_PARAERR : Parameter error
Include file	Srv_wdc.h		
Functional description	Whether to turn ON or OFF the LCD sleep mode is specified.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int mode; /* Variable for specifying whether to turn ON or OFF the rear LCD sleep mode */ int ret; /* Function's return value */ /** Parameter setup */ mode = ELIB_WDC_ON; /** Function call */ ret = Elib_WDC_OnOff_BackLcdSleep (mode);</pre>		

4.16 Battery level reference

4-16 Battery level reference			
Classification	Equipment Service		
Function	Battery level reference		
Symbol	Elib_WDC_Get_BattLvl		
Syntax	Int Elib_WDC_Get_BattLvl();		
Argument	Type	I/O	Description
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_BATTALM : Low-voltage alarm (unused) ELIB_WDC_BATTMIN : Lowest level ELIB_WDC_BATTLVL2 : Level 2 (unused) ELIB_WDC_BATTLVL3 : Level 3 ELIB_WDC_BATTLVL4 : Level 4 (unused) ELIB_WDC_BATTLVL5 : Level 5 (unused) ELIB_WDC_BATTMAX : Highest level ELIB_WDC_BATTOVER: Overvoltage alarm (unused) ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	The current battery level status is referenced. - The return value indicates the current battery level. - ELIB_WDC_BATTMAX (highest level) is always returned when the battery is being charged.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int ret; /* Function's return value */ /* Parameter setup */ /* Function call */ ret = Elib_WDC_Get_BattLvl();</pre>		

4.17 Earphone connection status reference

4-17 Earphone connection status reference			
Classification	Equipment Service		
Function	Earphone connection status reference		
Symbol	Elib_WDC_Get_EarPhone		
Syntax	int Elib_WDC_Get_EarPhone();		
Argument	Type	I/O	Description
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_EAR_ON : Earphone connected ELIB_WDC_EAR_OFF: Earphone not connected ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	The earphone connection status is referenced. - The return value indicates whether an earphone is connected to the terminal.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int ret; /* Function's return value */ /** Parameter setup */ /** Function call */ ret = Elib_WDC_Get_EarPhone();</pre>		

4.18 Earphone mode reference

4-18 Earphone mode reference			
Classification	Equipment Service		
Function	Earphone mode reference		
Symbol	Elib_WDC_Get_EarPhoneMode		
Syntax	int Elib_WDC_Get_EarPhoneMode();		
Argument	Type	I/O	Description
Return value	Type	I/O	Description
Ret	Int	O	ELIB_WDC_EAR : Earphone only ELIB_WDC_SOUND : Earphone + speaker ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	The earphone mode is referenced - The return value indicates the current earphone mode.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int ret; /* Function's return value */ /** Parameter setup */ /** Function call */ ret = Elib_WDC_Get_EarPhoneMode();</pre>		

4.19 Open/closed status reference

4-19 Open/closed status reference			
Classification	Equipment Service		
Function	Open/closed status reference		
Symbol	Elib_WDC_Get_OpenClose		
Syntax	int Elib_WDC_Get_OpenClose();		
Argument	Type	I/O	Description
Return value	Type	I/O	Description
Ret	Int	O	ELIB_WDC_OPEN : Open status ELIB_WDC_CLOSE : Closed status ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	Information about whether the mobile terminal is open or closed (if it is a folding type) is referenced. - The return value indicates whether the mobile terminal is open or closed.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int ret; /* Function's return value */ /** Parameter setup */ /** Function call */ ret = Elib_WDC_Get_OpenClose();</pre>		

4.20 Charger status reference

4-20 Charger status reference			
Classification	Equipment Service		
Function	Charger status reference		
Symbol	Elib_WDC_Get_Charge		
Syntax	int Elib_WDC_Get_Charge();		
Argument	Type	I/O	Description
Return value	Type	I/O	Description
Ret	Int	O	ELIB_WDC_CHARGE_OVER : External overvoltage detected ELIB_WDC_CHARGE_INF0 : Charger status "Charging" ELIB_WDC_CHARGE_INF1 : Charger status "Charge completed" ELIB_WDC_CHARGE_OFF : Charger not connected ELIB_WDC_CHARGE_TEMP_ERR : Abnormal temperature detected ELIB_WDC_CHARGE_BATT_ERR : Battery abnormality detected ELIB_WDC_CHARGE_OUTBATT_START: Constant-voltage mode ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	The charger status is referenced. - The return value indicates the current charger status.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre> int ret; /* Function's return value */ /** Parameter setup */ /** Function call */ ret = Elib_WDC_Get_Charge(); </pre>		

4.21 Manner mode ON/OFF reference

4-21 Manner mode ON/OFF reference			
Classification	Equipment Service		
Function	Manner mode ON/OFF reference		
Symbol	Elib_WDC_Get_Safety		
Syntax	int Elib_WDC_Get_Safety();		
Argument	Type	I/O	Description
Return value	Type	I/O	Description
Ret	Int	O	ELIB_WDC_SAFETY_ON : Manner mode ON ELIB_WDC_SAFETY_OFF : Manner mode OFF ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	<p>The manner mode ON/OFF status is referenced.</p> <ul style="list-style-type: none"> - The return value indicates whether the current manner mode is ON or OFF (user-specified value). * This function is intended to reference a set value. To acquire the current operation status, use "Manner mode operation status reference." 		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int ret; /* Function's return value */ /** Parameter setup */ /** Function call */ ret = Elib_WDC_Get_Safety();</pre>		

4.22 Manner mode setup reference

4-22 Manner mode setup reference			
Classification	Equipment Service		
Function	Manner mode setup reference		
Symbol	Elib_WDC_Get_SafetyMode		
Syntax	int Elib_WDC_Get_SafetyMode();		
Argument	Type	I/O	Description
Return value	Type	I/O	Description
Ret	Int	O	ELIB_WDC_SAFETY_SILENT : Expansion by PF vendor ELIB_WDC_SAFETY_VIB : Manner mode ELIB_WDC_SAFETY_ORG : Original manner mode ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	The manner mode setup is referenced - The return value indicates the type of the manner mode.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre> int ret; /* Function's return value */ /** Parameter setup */ /** Function call */ ret = Elib_WDC_Get_SafetyMode(); </pre>		

Use example	<p><Acquiring a melody-linked vibration pattern name and the corresponding vibration ID></p> <pre> _ELIB_WDC_VIB_TITLE Vib_DATA; /* Structure for holding a pattern name and vibration ID */ int ret; /* Function's return value */ ret = Elib_WDC_Get_VibTitle(- 4, &Vib_DATA); /* Vibration ID acquisition */ </pre>
-------------	---

4.24 Power ON status reference

4-24 Power ON status reference			
Classification	Equipment Service		
Function	Power ON status reference		
Symbol	Elib_WDC_Get_PowerOnStat		
Syntax	int Elib_WDC_Get_PowerOnStat();		
Argument	Type	I/O	Description
-	-	-	
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_PWRSW : Power SW-activated power ON /Battery coming off (allowable short power dip time elapsed) ELIB_WDC_PWRALM : Alarm-activated power ON ELIB_WDC_PWRWDT : Watchdog timer reset ELIB_WDC_PWRRST : Software reset ELIB_WDC_BATTOVER : Overvoltage alarm ELIB_WDC_HWRST : Hardware reset (recovery from short power dip) ELIB_WDC_PWRSET : Power source (battery or charger) attached ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	The power ON status is referenced.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int ret; /* Function's return value */ /** Parameter setup */ /** Function call */ ret = Elib_WDC_Get_PowerOnStat();</pre>		

4.25 Secret mode reference

4-25 Secret mode reference			
Classi	Equipment Service		
Function	Secret mode reference		
Symbol	Elib_WDC_Get_SecretMode		
Syntax	int Elib_WDC_Get_SecretMode();		
Argument	Type	I/O	Description
-	-	-	
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_NORMAL_MODE : Normal mode ELIB_WDC_SECRET_MODE : Secret mode ELIB_WDC_SECRET_SMODE: Secret-only mode ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	The secret mode is referenced.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int ret; /* Function's return value */ /** Parameter setup */ /** Function call */ ret = Elib_WDC_Get_SecretMode();</pre>		

4.26 Vibration pattern reference

4-26 Vibration pattern reference			
Classification	Equipment Service		
Function	Vibration pattern reference		
Symbol	Elib_WDC_Get_VibMode		
Syntax	int Elib_WDC_Get_VibMode(kind);		
Argument	Type	I/O	Description
-	-	-	
Kind	int	I	ELIB_WDC_VIB_VOICE : Incoming audiophone call setup ELIB_WDC_VIB_MAIL : Incoming e-mail setup ELIB_WDC_VIB_MSG_R : Incoming R-message setup ELIB_WDC_VIB_MSG_F : Incoming F-message setup ELIB_WDC_VIB_TV_VOICE : Incoming videophone call setup * A parameter error is assumed if any other setup is made.
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_VIB_PAT0 : Pattern No. 0 (OFF) ELIB_WDC_VIB_PAT1 : Pattern No. 1 (pattern 1) ELIB_WDC_VIB_PAT2 : Pattern No. 2 (pattern 2) ELIB_WDC_VIB_PAT3 : Pattern No. 3 (pattern 3) ELIB_WDC_VIB_PATMELO: Pattern No. 4 (linked with a melody) ELIB_WDC_NG : Abnormal end ELIB_WDC_PARAERR : Parameter error
Include file	srv_wdc.h		
Functional description	The vibration pattern setup is referenced. - To be specific, the call type-specific vibration pattern specified in "kind" is referenced. * The default value for each of the incoming audiophone call, e-mail, R-message, F-message, and videophone call is OFF.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre> <Referencing the vibration pattern set up for incoming e-mail> int kind; /* Variable for specifying an incoming call type for setup */ int ret; /* Function's return value */ /** Parameter setup */ kind = ELIB_WDC_VIB_MAIL ; /* Incoming e-mail setup */ </pre>		

	/** Function call **/
--	-----------------------

	ret = Elib_WDC_Get_VibMode(kind);
--	-----------------------------------

4.27 Manner mode operation status reference

4-27 Manner mode operation status reference			
Classification	Equipment Service		
Function	Manner mode operation status reference		
Symbol	Elib_WDC_Get_SafetyModeCtrl		
Syntax	int Elib_WDC_Get_SafetyModeCtrl();		
Argument	Type	I/O	Description
-	-	-	
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_SAFETY_SILENT: Expansion by PF vendor ELIB_WDC_SAFETY_VIB : Manner mode ELIB_WDC_SAFETY_ORG : Original manner ELIB_WDC_SAFETY_NORMAL: Manner mode OFF ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	The manner mode operation status (internal status) is referenced. - This function enables the operation status of the manner mode to be referenced.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre> int ActMode, OnOffMode; ActMode = Elib_WDC_Get_SafetyModeCtrl(); /* Current manner mode operation status */ OnOffMode = Elib_WDC_Get_Safety(); /* ON/OFF setup status */ if (ActMode == ELIB_WDC_SAFETY_NORMAL) { if (OnOffMode == ELIB_WDC_SAFETY_OFF) /* ; */ } </pre>		

4.28 Terminal lock(all feature lock) ON/OFF status reference

4-28 Terminal lock (all-feature lock) ON/OFF status reference			
Classification	Equipment Service		
Function	Terminal lock (all-feature lock) ON/OFF reference		
Symbol	Elib_WDC_Get_TrmLock		
Syntax	int Elib_WDC_Get_TrmLock();		
Argument	Type	I/O	Description
-	-	-	
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_TRM_ON : Terminal lock (all-feature lock) ON ELIB_WDC_TRM_OFF : Terminal lock (all-feature lock) OFF ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	The terminal lock (all-feature lock) ON/OFF is referenced. - The return value indicates whether the terminal lock (all-feature lock) is ON or OFF.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int ret; /* Function's return value */ /** Parameter setup */ /** Function call */ ret = Elib_WDC_Get_TrmLock();</pre>		

4.29 PIM lock ON/OFF status reference

4-29 PIM lock ON/OFF status reference			
Classification	Equipment Service		
Function	PIM lock ON/OFF reference		
Symbol	Elib_WDC_Get_PIMLock		
Syntax	int Elib_WDC_Get_PIMLock();		
Arg	Type	I/O	Description
-	-	-	
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_PIM_ON : PIM lock ON ELIB_WDC_PIM_OFF : PIM lock OFF ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	The PIM lock ON/OFF is referenced. - The return value indicates whether the PIM lock is ON or OFF.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int ret; /* Function's return value */ /** Parameter setup */ /** Function call */ ret = Elib_WDC_Get_PIMLock();</pre>		

4.30 Keyboard-activated dialing ban ON/OFF reference

4-30 Keypad-activated dialing ban ON/OFF reference			
Classification	Equipment Service		
Function	Keypad-activated dialing ban ON/OFF reference		
Symbol	Elib_WDC_Get_DialSend		
Syntax	int Elib_WDC_Get_DialSend();		
Argument	Type	I/O	Description
-	-	-	
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_DIAL_ON : Keypad-activated dialing ban ON ELIB_WDC_DIAL_OFF : Keypad-activated dialing ban OFF ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	The keypad-activated dialing ban ON/OFF status is referenced. - The return value indicates whether the keypad-activated dialing ban is ON or OFF.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int ret; /* Function's return value */ /** Parameter setup */ /** Function call */ ret = Elib_WDC_Get_DialSend();</pre>		

4.31 Battery pack detached-attached status reference

4-31 Battery pack detached-and-attached status reference			
Classification	Equipment Service		
Function	Battery pack detached-and-attached status reference		
Symbol	Elib_WDC_Get_BattPack_Off		
Syntax	int Elib_WDC_Get_BattPack_Off();		
Argument	Type	I/O	Description
-	-	-	
Ret	int	O	ELIB_WDC_BATTOFF_YES : The battery pack was detached and attached again. ELIB_WDC_BATTOFF_NO : The battery pack is kept attached. ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	Information about whether the battery pack was detached and attached again is referenced.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int ret; /* Function's return value */ /** Parameter setup */ s /** Function call */ ret = Elib_WDC_Get_BattPack_Off();</pre>		

4.32 Operation mode acquisition processing

4-32 Operation mode acquisition processing			
Classification	Equipment Service		
Function	Operation mode acquisition processing		
Symbol	Elib_WDC_Get_Actmode_maw		
Syntax	int Elib_WDC_Get_Actmode_maw();		
Argument	Type	I/O	Description
-	-	-	
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_TEST : Test mode ELIB_WDC_NORMAL : Normal mode ELIB_WDC_SPECIAL : Special mode ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	Information about the operation mode is acquired.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int ret; /* Function's return value */ /** Function call */ ret = Elib_WDC_Get_Actmode_maw();</pre>		

4.33 IMEI reference

4-33 IMEI reference			
Classification	Equipment Service		
Function	IMEI reference		
Symbol	Elib_WDC_Get_IMEI		
Syntax	int Elib_WDC_Get_IMEI(data);		
Argument	Type	I/O	Description
-	-	-	
Data	_ELIB_WDC_IMEI *	O	Pointer to a structure for holding IMEI information See "Functional description" for detailed descriptions about the structure.
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_PARAERR : Parameter error
Include file	srv_wdc.h		
Functional description	<p>IMEI information is referenced, and the resulting ASCII character string is stored to the _ELIB_WDC_IMEI structure.</p> <p>< _ELIB_WDC_IMEI information structure></p> <pre>typedef struct tagELIB_WDC_IMEI { unsigned char tac[6]; /* Type Approval Code */ unsigned char fac[2]; /* Final Assembly Code */ unsigned char snr[6]; /* Serial Number */ unsigned char spare[1]; /* Check Digit */ } _ELIB_WDC_IMEI;</pre>		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int ret; /* Function's return value */ _ELIB_WDC_IMEI data; /* Structure for holding IMEI information */ /** Function call */ ret = Elib_WDC_Get_IMEI(&data);</pre>		

4.34 USB connection status reference

4-34 USB connection status reference			
Classification	Equipment Service		
Function	USB connection status reference		
Symbol	Elib_WDC_Get_USBInfo		
Syntax	int Elib_WDC_Get_USBInfo();		
Argument	Type	I/O	Description
-	-	-	
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_USB_ON : USB connected ELIB_WDC_USB_OFF: USB not connected ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	The USB connection status is referenced. - The return value indicates whether the USB is connected.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int ret; /* Function's return value */ /** Function call */ ret = Elib_WDC_Get_USBInfo();</pre>		

4.35 UIM insertion status reference

4-35 UIM insertion status reference			
Classification	Equipment Service		
Function	UIM insertion status reference		
Symbol	Elib_WDC_Get_UIMStatus		
Syntax	int Elib_WDC_Get_UIMStatus();		
Argument	Type	I/O	Description
-	-	-	
Return value	Type	I/O	Description
Ret	int	O	UIM card insertion status (See "Functional description."): ELIB_WDC_OK ELIB_WDC_CONNECT_NG ELIB_WDC_CONNECT_OFF ELIB_WDC_CONNECT_ERR ELIB_WDC_FAIL_CARD ELIB_WDC_NG
Include file	srv_wdc.h		
Functional description	Function for referencing the UIM card insertion status. - The return value indicates whether the UIM card has been inserted. <UIM card insertion status> ELIB_WDC_OK : UIM connected normally (connected with no error message received) ELIB_WDC_CONNECT_NG: UIM abnormality (UIM failure occurred) ELIB_WDC_CONNECT_OFF: UIM not connected (no UIM inserted*) ELIB_WDC_CONNECT_ERR: UIM unusable ELIB_WDC_FAIL_CARD : Wrong UIM card inserted (there is a non-W-CDMA (3GPP standard) application in the UICC or there is no application at all in the UICC.*) ELIB_WDC_NG : Abnormal end * Set up at power-ON time.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int ret; /* Function's return value */ /** Function call */ ret = Elib_WDC_Get_UIMStatus();</pre>		

4.36 Charge start/end confirmatory sound status reference

4-36 Charge start/end confirmatory sound setup status reference			
Classification	Equipment Service		
Function	Charge start/end confirmatory sound setup status reference		
Symbol	Elib_WDC_Get_ChargeSound		
Syntax	int Elib_WDC_Get_ChargeSound();		
Argument	Type	I/O	Description
-	-	-	
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_CHARGESND_ON : Set up ELIB_WDC_CHARGESND_OFF: Released ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	Information about whether to generate confirmatory sound at the start and completion of battery charge is referenced.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int ret; /* Function's return value */ /** Function call */ ret = Elib_WDC_Get_ChargeSound();</pre>		

4.37 Vibrator operation status reference

4.37 Vibrator operation status reference			
Classification	Equipment Service		
Function	Vibrator operation status reference		
Symbol	Elib_WDC_Get_ActVib		
Syntax	int Elib_WDC_Get_ActVib(vib_pat);		
Argument	Type	I/O	Description
-	-	-	
vib_pat	int *	O	<p>Pointer to an area for holding a vibration pattern.</p> <p>The return value indicates the current vibration pattern when vibration is in progress (valid only when vibration is in progress).</p> <p>ELIB_WDC_VIB_PAT1 : Vibration with pattern 1 in progress ELIB_WDC_VIB_PAT2 : Vibration with pattern 2 in progress ELIB_WDC_VIB_PAT3 : Vibration with pattern 3 in progress ELIB_WDC_VIB_PATMELO: Vibration linked with a melody in progress</p> <p>* Specify NULL if it is unnecessary to identify the vibration pattern currently being used for vibration in progress. If it is impossible for Service to identify the vibration pattern currently being used for vibration in progress, however, *vib_pat = NULL is returned.</p>
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_VIB_OFF : Vibration not in progress ELIB_WDC_VIB_ON : Vibration in progre ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	- The current vibrator operation status is referenced.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int ret; /* Function's return value */ int vib_pat; /* Vibration pattern */ /** Function call */ ret = Elib_WDC_Get_ActVib(&vib_pat);</pre>		

4.38 Banned-operation setup status reference

4-38 Banned-operation setup status reference

Classification	Equipment Service																
Function	Banned-operation setup status reference																
Symbol	Elib_WDC_Get_Prohibit_Status																
Syntax	int Elib_WDC_Get_Prohibit_Status(pro_sts);																
Argument	Type	I/O	Description														
pro_sts	int *	O	Banned-operation setup status See "Functional description."														
Return value	Type	I/O	Description														
Ret	int	O	ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_PARAERR : Parameter error														
Include file	srv_wdc.h																
Functional description	<p>Information about the currently banned operations is referenced.</p> <p>- The following setup statuses are returned.</p> <div style="display: flex; align-items: center; margin-top: 10px;"><div style="margin-right: 20px;">31 10 9 8 7 6 5 4 3 2 1 0</div><table border="1" style="border-collapse: collapse; text-align: center;"><tr><td style="width: 20px; height: 30px;"></td><td style="width: 20px; height: 30px;"></td><td style="width: 20px; height: 30px;"></td><td style="width: 20px; height: 30px;"></td><td style="width: 20px; height: 30px;"></td><td style="width: 20px; height: 30px;"></td><td style="width: 20px; height: 30px;"></td><td style="width: 20px; height: 30px;"></td><td style="width: 20px; height: 30px;"></td><td style="width: 20px; height: 30px;"></td><td style="width: 20px; height: 30px;"></td><td style="width: 20px; height: 30px;"></td><td style="width: 20px; height: 30px;"></td><td style="width: 20px; height: 30px;"></td></tr></table><div style="margin-left: 10px;">Type: int</div></div> <p style="margin-top: 10px;">Bits 7 to 31 ----- Unused</p> <p>Bit 6 ----- 1: ADL ON (started) 0: ADL OFF (not started)</p> <p>Bit 5 ----- 1: Keypad-activated dialing ban setup ON 0: Keypad-activated dialing ban setup OFF</p> <p>Bit 4 ----- 1: Phone book-activated dialing lock setup ON 0: Phone book-activated dialing lock setup OFF *1</p> <p>Bit 3 ----- 1: Application lock setup ON 0: Application lock setup OFF *1</p> <p>Bit 2 ----- 1: PIM lock setup ON 0: PIM lock setup OFF</p> <p>Bit 1 ----- 1: Key operation disable setup ON 0: Key operation disable setup OFF *1</p> <p>Bit 0 ----- 1: Terminal lock setup ON 0: Terminal lock setup OFF</p> <p style="margin-top: 10px;">*1 The key operation disable, phone book-activated dialing lock, and application lock feature bits are kept OFF, because the features have been deleted.</p>																
Related message	None																
Necessary procedure																	

Note	
Prohibition	None
Use example	<pre>int pro_sts; /* Banned operation setup status */ int ret; /* Function's return value */ /** Function call */ ret = Elib_WDC_Get_Prohibit_Status(&pro_sts);</pre>

4.39 Equipment Service message reference

4-39 Equipment Service message reference			
Classification	Equipment Service		
Function	Equipment Service message reference		
Symbol	Elib_WDC_Get_Message		
Syntax	int Elib_WDC_Get_Message(type, buflen, buff);		
Argument	Type	I/O	Description
Type	int	I	Message types ELIB_WDC_MSG_BATTALM Low-voltage alarm message ELIB_WDC_MSG_CHARGEOVER External overvoltage message ELIB_WDC_MSG_UIMNG UIM abnormality message ELIB_WDC_MSG_UIMOFF UIM not connected message ELIB_WDC_MSG_UIMERR UIM unusable message ELIB_WDC_MSG_UNCORR Wrong card inserted message
Buflen	size_t	I	Size of the "buffer" for acquiring character strings * The return value is ELIB_WDC_NG if the specified buffer size is insufficient for holding all the character strings involved.
Buff	unsigned char *	I/O	Pointer to a buffer for holding character strings * The buffer shall be secured by the APL that uses it.
Return value	Type	I/O	Description
Ret	int	O	Normal end: The specified character string length is returned. Abnormal end: ELIB_WDC_NG Parameter error: ELIB_WDC_PARAERR
Include file	srv_wdc.h		
Functional description	The wording corresponding to the message type is stored. - Wording (characters) is in SJIS format. * The maximum message length is 60 bytes.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre> unsigned char buff[255]; /* Message acquisition buffer */ int ret; /* Function's return value */ /* Function call */ ret = Elib_WDC_Get_Message(, ELIB_WDC_MSG_BATTALM, sizeof(buff), buff); </pre>		

4.40 User-specified item reset

4-40 User-specified item reset			
Classification	Equipment Service		
Function	User-specified item reset		
Symbol	Elib_WDC_UsrSetReset		
Syntax	Int Elib_WDC_UsrSetReset(mode);		
Argument	Type	I/O	Description
Mode	int	I	Specified reset mode ELIB_WDC_NORMAL_RESET: Normal reset
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_PARAERR : Parameter error
Include file	Srv_wdc.h		
Functional description	Calling this function enables each user-specified item (feature) to be reset. * Each responsible block shall perform actual reset processing.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre> int ret; /* Function's return value */ int mode; /* Specified reset mode */ /** Parameter setup */ mode = ELIB_WDC_NORMAL_RESET; /** Function call */ ret = Elib_WDC_UsrSetReset(mode); </pre>		

4.41 Forced power-OFF notice

4-41 Forced power-OFF notice			
Classification	Equipment Service		
Function	Forced power-OFF notice		
Symbol	Elib_WDC_PowerOff		
Syntax	int Elib_WDC_PowerOff();		
Argument	Type	I/O	Description
-	-	-	
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end
Include file	srv_wdc.h		
Functional description	A forced power-OFF notice is sent to the MAW.		
Related message	None		
Necessary procedure			
Note			
Prohibition	None		
Use example	<pre>int ret; /* Function's return value */ /** Function call */ ret = Elib_WDC_PowerOff();</pre>		

4.42 Personal data setup/reference processing

4-42 Personal data setup/reference processing			
Classification	Equipment Service		
Function	Personal data setup/reference processing		
Symbol	Elib_WDC_OpeProfileSetRef		
Syntax	int Elib_WDC_OpeProfileSetRef(mode, pro_data);		
Argument	Type	I/O	Description
Mode	unsigned char	I	Specifying whether to setup or reference ELIB_FUNCSET: Setup ELIB_FUNCREF: Reference
pro_data	_ELIB_WDC_PROFILE *	I/O	Pointer to the profile data structure
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_PARAERR : Parameter error
Include file	srv_wdc.h		
Functional description	<p>User-edited telephone numbers, names, and e-mail addresses are set up or referenced.</p> <p>This interface cannot reference telephone number 1 (telephone number assigned to the terminal of interest). Data Switch Service supports an interface for telephone number 1 reference.</p> <p>The maximum number of characters that can be specified for each of names and their furigana names is 32 bytes for each full name.</p> <p>The telephone function interface (ELIB) does not check the character string length. It shall be done on the APL side.</p> <pre> typedef struct tagWDC_PROFILE { unsigned char name[ELIB_WDC_KANJIMAX+1]; /* Name */ unsigned char name_mei[ELIB_WDC_KANJI_MEI_MAX+1]; /* First name */ unsigned char kana[ELIB_WDC_KANAMAX+1]; /* Furigana name */ unsigned char kana_mei[ELIB_WDC_KANA_MEI_MAX+1]; /* Furigana first name */ unsigned char memo[ELIB_WDC_MEMO_MAX+1]; /* Memo */ unsigned char picture; /* Whether there is picture data */ _ELIB_WDC_TELNO dial[3]; /* Telephone numbers (2 to 4), icon */ _ELIB_WDC_MAILADR mail_adr[3]; /* E-mail address, icon */ _ELIB_WDC_ZIPADDRESS address; /* Postal code and street address */ _ELIB_WDC_SIPADR sip_adr[2]; /*Unused*/ unsigned char imode_url[2][ELIB_WDC_URLMAX+1]; /* URL (for i-mode) */ unsigned char wlan_url[2][ELIB_WDC_URLMAX+1]; /*Unused*/ } _ELIB_WDC_PROFILE; </pre> <p>* Names, their furigana names, telephone numbers, and e-mail addresses shall be suffixed with NULL.</p>		

```

typedef struct tagWDC_TELNO
{
    unsigned char tel_no[ELIB_WDC_DIALMAX+1]; /* Telephone number */
    unsigned short tel_icon; /* Telephone number icon */
} _ELIB_WDC_TELNO;

typedef struct tagWDC_MAILADR
{
    unsigned char mail_adr[ELIB_WDC_MAILMAX+1]; /* E-mail address */
    unsigned short adr_icon; /* E-mail address icon */
} _ELIB_WDC_MAILADR ;
    * The e-mail address shall include "@"

typedef struct tagWDC_ZIPADDRESS
{
    unsigned char post [ELIB_WDC_ZIP_MAX+1]; /* Postal code */
    unsigned char address[ELIB_WDC_ADDRESS_MAX+1];
                                                    /* Street address */
} _ELIB_WDC_ZIPADDRESS;

#define ELIB_WDC_KANJIMAX    32
                                /* Upper limit to the kanji (family name) data length */
#define ELIB_WDC_KANJI_MEI_MAX    32
                                /* Upper limit to the kanji (first name) data length */
#define ELIB_WDC_KANAMAX    32
                                /* Upper limit to the furigana (family name) data length */
#define ELIB_WDC_KANA_MEI_MAX    32
                                /* Upper limit to the furigana (first name) data length */
#define ELIB_WDC_DIALMAX    26 /* ASCII type dial data length */
#define ELIB_WDC_MAILMAX    50
                                /* Maximum number of bytes in each e-mail address */
#define ELIB_WDC_ZIP_MAX    7
                                /* Upper limit to the postal code data length */
#define ELIB_WDC_ADDRESS_MAX    93
                                /* Upper limit to the street address data length */
#define ELIB_WDC_MEMO_MAX    100
                                /* Upper limit to the memo data length */
#define ELIB_WDC_PICT_OFF    0 /* There is no picture data. */
#define ELIB_WDC_PICT_ON    1 /* There is picture data. */
#define ELIB_WDC_URLMAX    256
                                /* Upper limit to the URL data length */

    [About picture data]
    (When setting up)
    To set up picture data, set "Whether there is picture data (picture)" with "There is picture
data (ELIB_WDC_PICT_ON)."
    (When deleting)

```

	To delete picture data, set "Whether there is picture data (picture) " with "There is no picture data (ELIB_WDC_PICT_OFF). "
Related message	None
Necessary procedure	
Note	
Prohibition	None
Use example	<pre> int ret; /* Function's return value */ _ELIB_WDC_PROFILE pro_data; /* Structure for folding profile data */ /** Function call */ ret = Elib_WDC_OpeProfileSetRef(ELIB_FUNCREF, &pro_data); </pre>

4.43 Personal picture data setup/reference processing

4-43 Personal picture data setup/reference processing			
Classification	Equipment Service		
Function	Personal picture data setup/reference processing		
Symbol	Elib_WDC_OpeProfile_Picture_SetRef		
Syntax	int Elib_WDC_OpeProfile_Picture_SetRef(mode, data);		
Argument	Type	I/O	Description
Mode	unsigned char	I	Specifying whether to set up or reference ELIB_FUNCSET : Setup ELIB_FUNCREF : Reference
Data	_RES_IMG_INFO *	I	Picture data structure pointer
Return value	Type	I/O	Description
Ret	int	O	ELIB_WDC_OK : Normal end ELIB_WDC_NG : Abnormal end ELIB_WDC_PRMERR : Parameter error
Include file	srv_wdc.h		
Functional description	<p>Picture data registered as personal data is referenced or set up.</p> <p>[Referencing] If you need to acquire picture data, do it, using this interface. # The caller shall previously secure a picture data storage area having a sufficient size and set its pointer.</p> <pre>/* Structure for registering and setting picture display data */ typedef struct tagRES_IMG_INFO { _RES_IMG_HEADINFO head; /* Image header (see the picture display data header structure) */ unsigned char *data; /* Picture data */ } _RES_IMG_INFO ;</pre> <p># The maximum size of one picture data item that can be registered is 100 KB. The _RES_IMG_INFO structure definition is a telephone function resource management library interface and it becomes necessary to include the files of the telephone function resource management library.</p> <p>[When setting up] Register picture data to the personal data area. This function writes the picture data over the existing one (if there is) regardless of what the previous status is (no matter whether picture data has been registered).</p> <p>[when deleting] Specifying ELIB_WDC_PICT_OFF as the picture data presence/absence flag of "Personal data</p>		

	setup/reference_processing (Elib_WDC_OpeProfileSetRef) " causes no-picture data status (equivalent to deletion).
Related message	None
Necessary procedure	
Note	
Prohibition	None
Use example	<pre> int ret; /* Function's return value */ _RES_IMG_INFO data; /* Structure for registering and setting up picture display data */ /** Function call */ ret = Elib_WDC_OpeProfile_Picture_SetRef(ELIB_FUNCREF, &data); </pre>

5. Schedule Service

5.1 Starat requesting Service event notifications

Classification	Schedule Service supporting function		
Description	Start requesting Schedule Service event notifications	Function name	Elib_SCH_Request
Functional overview	<ul style="list-style-type: none"> - Applications which are in need of Schedule Service events can call this function to request the relative notifications. - Notifications requests about Schedule Service events remain registered until " Stop requesting Schedule Service event notifications" is executed to cancel these requests. - Notifications about schedule events registered by other applications (by using the "Register schedule event" and/or "Register wake-up alarm" function) can be posted by using this function to make a request for such events. - The type of the callback function on occurrence of an event is as follows: void func(MsbObject *client, MsbObject *server, void *sendData, void *recvDate, void *data, MsbEnvironment *ev); * MSB is used for the Event notifications. - When an event notification happens (a callback function is called), the event data are stored in sendData. 		
Include file	srv_sch.h msb/msb.h (provided by msb)		
Prototype	int Elib_SCH_Request(Ap_ID, event, func);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
event	int	I	Notification events Time change notification event : ScheduleNotify_Interval Clock change notification event : ScheduleNotify_ChgCal Regular alarm event : ScheduleNotify_Normal Wake-up alarm event : ScheduleNotify_WakeUp ToDo alarm event : ScheduleNotify_ToDo Java schedule event : ScheduleNotify_Java

			Deletion of multiple items completed notification event : ScheduleNotify_Proc ADL reserved time event : ScheduleNotify_ADL
func	MsbFunc	I	Callback function (called when the event occurs)
Return value	Type	I/O	Description
Ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter: ELIB_SCH_PARAM_NG
Remark			
<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications. (For details, refer to "MSB External API Specifications.")</p> <p>Notes:</p> <ol style="list-style-type: none"> To reference information in event data, make a cast to each data type. To receive notifications of the event specified by this function, call the "Start posting alarm notifications" function to post that the application is ready to receive notification events when it starts up. However, the "time change notification" event is posted when the system time is updated after this function is called. Also, the clock change notification event is posted when the date and time is changed after the "Set system date/time" function is executed. 			

5.2 Stop requesting Schedule Service event notifications

Classification	Schedule Service supporting function		
Description	Stop requesting Schedule Service event notifications	Function name	Elib_SCH_Cancel
Functional overview	<p>This function stops the request for Schedule Service event notifications registered by "Start requesting Schedule Service event notifications."</p>		
Include file	srv_sch.h		
Prototype	int Elib_SCH_Cancel(Ap_ID, event);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
event	int	I	Notification events Time change notification event : ScheduleNotify_Interval Clock change notification event : ScheduleNotify_ChgCal Regular alarm event : ScheduleNotify_Normal Wake-up alarm event : ScheduleNotify_WakeUp ToDo alarm event : ScheduleNotify_ToDo Java schedule event : ScheduleNotify_Java Deletion of multiple items completed notification event : ScheduleNotify_Proc ADL reserved time event: ScheduleNotify_ADL
Return value	Type	I/O	Description
Ret	int	O	Successful: ELIB_SCH_OK Unsuccessful: ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered : ELIB_SCH_NODATA
Remark	<p>Necessary procedure: Create a MSB object by using MsbObjectCreate in respective applications.</p>		

alarm sound, use the ring interface of Sound System Service.

- For the icon data setting, specify any of the values indicated in "5. List of Icon Data Specifiers." Schedule Service only registers, reads, and searches for data delivered to it, thus the application needs to associate it with read data.
- To receive notifications about an event specified by this function, the application needs to make a call to the "A-19 Start posting alarm notifications" function when it starts up to inform that the application is ready to process the event.
- Even if the alarm type is set to "Alarm turned off" for the event, the application is notified of the event.
- The time for prior alarm notice is returned as it is regardless of the alarm type.

Include file	srv_sch.h		
Prototype	int Elib_SCH_Normal_Set(Ap_ID, Sch_Data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Sch_Data	_ELIB_SCH_DATA *	I/O	Schedule data to be registered
Return value	Type	I/O	Description
Ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered data : ELIB_SCH_NODATA Max number of registered events reached : ELIB_SCH_MAXDATA Not supported year : ELIB_SCH_OVERFLOW Weekly specified setting out of scope : ELIB_SCH_WEEKOVERFLOW
Remark			

Necessary procedure:

Create a MSB object by using MsbObjectCreate in respective applications.

Notes:

1. When the "date and time of start" and "end date and time" are set to different days for a schedule, the schedule is regarded that it is repeated in the term between the "date and time of start" and the "end date and time." However, the repeated events are regarded as of a single schedule.

The table below shows the details of the "Same time/Same type" and "Same time/One-off priority":

Return value	Schedule type		Condition
Same time/Same type	One-off	One-off	Year, month, and day = Year, month, and day
	Weekly	Weekly	The ranges of the schedules are overlapping and the schedules have one or more same days of the week specified.
	Daily	Weekly	The ranges of the schedules are overlapping.
	Daily	Daily	The ranges of the schedules are overlapping.
Same time/One-off priority	One-off	Daily	The ranges of the schedules are overlapping and Date of one-off \geq Date of Daily
	One-off	Weekly	The ranges of the schedules are overlapping. Day of the week of one-off = Day of the week specified, and Date of one-off \geq Date of Daily

If there are a "one-off" schedule and schedule "repeated (daily/weekly)" which are set to the same time, either of the following functions is used for the return value:

- When the schedule to be checked for date/time is "one-off" \rightarrow "Same time/Same type"
- When the schedule to be checked for date/time is "Repeated" \rightarrow "Same time/One-off priority"
- This function sets the data to be checked into the schedule data structure represented by the schedule to be checked for date/time. The result is put into the said structure.

Include file	srv_sch.h		
Prototype	int Elib_SCH_Same_Time_Check(Ap_ID, Sch_Data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Sch_Data	ELIB_SCH_DATA *	I/O	Schedule to be checked for date/time
Return value	Type	I/O	Description
Ret	int	O	Registerable : ELIB_SCH_OK Same time/One-off priority : ELIB_SCH_SINGLEPRI Same time/Same type : ELIB_SCH_SAMETYPE Not supported year : ELIB_SCH_OVERFLOW Weekly specified setting out of scope

			: ELIB_SCH_WEEKOVERFLOW
		Incorrect parameter	: ELIB_SCH_PARAM_NG
		Unsuccessful	: ELIB_SCH_NG
Remark			
Necessary procedure: Create a MSB object by using MsbObjectCreate in respective applications.			

5.5 Delete all schedules designated before specified date/time

Classification	Schedule Service supporting function																										
Description	Delete all schedules designated before specified date/time	Function name	Elib_SCH_PastSchedule_Reset																								
Functional overview	<div><ul style="list-style-type: none">- This function deletes data of all schedules designated before a specified target date.- The schedule attribute allows to narrow down schedules to be deleted.- The application calls this function after having set the target date into the date and time data structure.</div> <div>Target date and time (date and time data structure)</div> <table><tr><td>Data item name</td><td>Value</td><td>Remark</td></tr><tr><td>Year</td><td>Christian era</td><td>Christian year in the supported calendars</td></tr><tr><td>Month</td><td>1 to 12</td><td></td></tr><tr><td>Day</td><td>1 to 31</td><td></td></tr><tr><td>Day of the week</td><td></td><td>Unused</td></tr><tr><td>Hour</td><td></td><td>Unused</td></tr><tr><td>Minute</td><td></td><td>Unused</td></tr><tr><td>Second</td><td></td><td>Unused</td></tr></table> <div><ul style="list-style-type: none">- For any schedule to be repeated (schedules with the schedule type set to daily or weekly), its dates after the target date of deletion are re-set.- This function handles the dates of schedule data regardless of the time for prior notice offset.</div>			Data item name	Value	Remark	Year	Christian era	Christian year in the supported calendars	Month	1 to 12		Day	1 to 31		Day of the week		Unused	Hour		Unused	Minute		Unused	Second		Unused
Data item name	Value	Remark																									
Year	Christian era	Christian year in the supported calendars																									
Month	1 to 12																										
Day	1 to 31																										
Day of the week		Unused																									
Hour		Unused																									
Minute		Unused																									
Second		Unused																									
Include file	srv_sch.h																										
Prototype	int Elib_SCH_PastSchedule_Reset(Ap_ID, Class, Datetime);																										
Argument	Type	I/O	Description																								
Ap_ID	unsigned int	I	Application ID																								
Class	int	I	Schedule attribute: Data with unspecified secret setting only																								

			: ELIB_SCH_PUBLIC Data with specified secret setting only : ELIB_SCH_SECRET Irrelevant to secret setting : ELIB_SCH_ALLDATA
Datetime	_ELIB_SCH_CAL_DATA *	I	Target date and time
Return value	Type	I/O	Description
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.6 Read schedule data by category

Classification	Schedule Service supporting function		
Description	Read schedule data by category	Fuction name	Elib_SCH_Category_Read
Functional overview	<ul style="list-style-type: none"> - This function sorts data of schedules registered with a specified category in chronological order then reads them one by one. - This function sets read conditions into the schedule data structure represented by the schedule data readout area. The result is put into the said structure. (For details about the structure, see "4. List of Structures.") - This function lists schedules matching conditions specified in the initial call to it (made with 0 set to the listing number argument). In the second or later call, specifying the position of a schedule in the listing number argument allows the schedule data to be read. Note that the initial call returns the schedule at the top of the list. - This function outputs schedules not to be repeated (one-off) as they are. This function outputs schedules to be repeated (daily/weekly) calculating the next alarm date and time according to the date and time on the moment. - This function sorts schedules in chronological order using the alarm date and time as the key. - This function sorts schedules to be alarmed on the same date and time in order of registration. 		
Include file	srv_sch.h		
Prototype	int Elib_SCH_Category_Read(Ap_ID, No, List_Data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
No	int	I	Listing number
List_Data	_ELIB_SCH_DATA *	I/O	schedule data readout area
Return value	Type	I/O	Description
ret	int	O	Successful: Number of schedules in the specified category Unsuccessful : ELIB_SCH_NG

			Unregistered : ELIB_SCH_NODATA Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p> <p>Notes:</p> <ol style="list-style-type: none">1. The information put in the list by the initial call comes from static areas of functions. Therefore, the caller must note that an initial call made before it reads all schedule data modifies static areas.2. The alarm date and time does not include the time for prior notice.		

5.7 Read the number of registered schedules

Classification	Schedule Service supporting function		
Description	Read the number of registered schedules	Function name	Elib_SCH_NumOfRegisteredSchedule_Read
Functional overview	<ul style="list-style-type: none"> - This function gets the number of already registered schedules. - The schedule attribute allows to narrow down schedules to be deleted. 		
Include file	srv_sch.h		
Prototype	int Elib_SCH_NumOfRegisteredSchedule_Read(Ap_ID, Class);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Class	int	I	Schedule attribute: Data with unspecified secret setting only : ELIB_SCH_PUBLIC Data with specified secret setting only : ELIB_SCH_SECRET Irrelevant to secret setting : ELIB_SCH_ALLDATA
Return value	Type	I/O	Description
ret	int	O	Successful : Number of schedules registered Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	Necessary procedure: Create a MSB object by using MsbObjectCreate in respective applications.		

5.8 Read the number of schedules registered to specified date

Classification	Schedule Service supporting function																										
Description	Read the number of schedules registered to specified date	Function name	Elib_SCH_DayPlan_Read																								
Functional overview	<div><ul style="list-style-type: none">- This function gets the number of schedules registered to a specified date.- The type selection allows to narrow down schedules to be deleted.- The schedule attribute allows to narrow down schedules to be deleted.- This function sets the year, month, and day of the date from which the number of already registered schedules is to be acquired until the target date (date and time data structure). (For details about the structure, see "4. List of Structures.")</div> <div>Target date (date and time data structure)</div> <table><tr><td>Data item name</td><td>Value</td><td>Remark</td></tr><tr><td>Year</td><td>Christian era</td><td>Christian year in the supported calendars</td></tr><tr><td>Month</td><td>1 to 12</td><td></td></tr><tr><td>Day</td><td>1 to 31</td><td></td></tr><tr><td>Day of the week</td><td></td><td>Unused</td></tr><tr><td>Hour</td><td></td><td>Unused</td></tr><tr><td>Minute</td><td></td><td>Unused</td></tr><tr><td>Second</td><td></td><td>Unused</td></tr></table>			Data item name	Value	Remark	Year	Christian era	Christian year in the supported calendars	Month	1 to 12		Day	1 to 31		Day of the week		Unused	Hour		Unused	Minute		Unused	Second		Unused
Data item name				Value	Remark																						
Year				Christian era	Christian year in the supported calendars																						
Month				1 to 12																							
Day				1 to 31																							
Day of the week		Unused																									
Hour		Unused																									
Minute		Unused																									
Second		Unused																									
Include file	srv_sch.h																										
Prototype	int Elib_SCH_DayPlan_Read(Ap_ID, Mode, Class, Datetime);																										
Argument	Type	I/O	Description																								
Ap_ID	unsigned int	I	Application ID																								
Mode	int	I	Type All of the specified date : ELIB_SCH_DAYALL A.M. : ELIB_SCH_AM P.M. : ELIB_SCH_PM																								

			Days before the specified date : ELIB_SCH_BEFORE
Class	int	I	Schedule attribute: Data with unspecified secret setting only : ELIB_SCH_PUBLIC Data with specified secret setting only : ELIB_SCH_SECRET Irrelevant to secret setting : ELIB_SCH_ALLDATA
Datetime	_ELIB_SCH_CAL_ DATA *	I	Target date * The day of the week, hour, and minute do not need to be specified.
Return value	Type	I/O	Description
ret	int	O	Successful : Number of schedules registered Unsuccessful : ELIB_SCH_NG Unregistered : ELIB_SCH_NODATA Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.9 Search for schedule not alarmed

Classification	Schedule Service supporting function		
Description	Search for not alarmed schedule	Function name	Elib_SCH_ActiveAlmSearch
Functional overview	<ul style="list-style-type: none"> - This function searches for a regular schedule registered but not alarmed. - This function handles any schedule with the repetition setting specified (as far as the date is within the end date and time the calendar can support). - The schedule attribute allows to narrow down schedules to be searched for. - The calendared alarm date and time is used. 		
Include file	srv_sch.h		
Prototype	int Elib_SCH_ActiveAlmSearch(Ap_ID, Class);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Class	int	I	Schedule attribute: Data with unspecified secret setting only : ELIB_SCH_PUBLIC Data with specified secret setting only : ELIB_SCH_SECRET Irrelevant to secret setting : ELIB_SCH_ALLDATA
Return value	Type	I/O	Description
ret	int	O	Found (those on and after the present time) : ELIB_SCH_ALM_TODAY Found (those in future only) : ELIB_SCH_ALM_AFTER Not found : ELIB_SCH_NODATA Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark			
Necessary procedure:			

Create an MSB object by using `MsbObjectCreate` in respective applications.

Note:

The calendared alarm date and time includes the time for prior notice.

5.10 Read number of days in specified year/month

Classification	Schedule Service supporting function																										
Description	Read number of days in specified year/month	Function name	Elib_SCH_NumOfDays_Read																								
Functional overview	<div>- This function gets the number of days in a month of a specified year.</div> <div>- This function returns (in the date and time data structure) the number of days in the specified (target) year and month (For details about the structure, see "4. List of Structures.")</div> <div>Target year and month (date and time data structure)</div> <table><tr><td>Data item name</td><td>Value</td><td>Remark</td></tr><tr><td>Year</td><td>Christian era</td><td>Christian year in the supported calendars</td></tr><tr><td>Month</td><td>1 to 12</td><td></td></tr><tr><td>Day</td><td></td><td>Unused</td></tr><tr><td>Day of the week</td><td></td><td>Unused</td></tr><tr><td>Hour</td><td></td><td>Unused</td></tr><tr><td>Minute</td><td></td><td>Unused</td></tr><tr><td>Second</td><td></td><td>Unused</td></tr></table>			Data item name	Value	Remark	Year	Christian era	Christian year in the supported calendars	Month	1 to 12		Day		Unused	Day of the week		Unused	Hour		Unused	Minute		Unused	Second		Unused
Data item name				Value	Remark																						
Year	Christian era	Christian year in the supported calendars																									
Month	1 to 12																										
Day		Unused																									
Day of the week		Unused																									
Hour		Unused																									
Minute		Unused																									
Second		Unused																									
Include file	srv_sch.h																										
Prototype	int Elib_SCH_NumOfDays_Read(Ap_ID, Datetime);																										
Argument	Type	I/O	Description																								
Ap_ID	unsigned int	I	Application ID																								
Datetime	_ELIB_SCH_CAL_D ATA *	I	Target year and month																								
Return value	Type	I/O	Description																								
ret	int	O	Successful : Number of days in the specified month (28, 29, 30, or 31) Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG																								
Remark																											

Necessary procedure:

Create a MSB object by using MsbObjectCreate in respective applications.

5.11 Read day of week of specified date

Classification	Schedule Service supporting function																																								
Description	Read day of week of specified date	Function name	Elib_SCH_Week_Read																																						
Functional overview	<ul style="list-style-type: none"> - This function gets the day of the week of a specified date. - This function sets the date for which the day of the week is to be acquired into the date and time data structure represented by the target date. The result is put into the day of the week of the said structure. <p>Target date (date and time data structure)</p> <table> <tr> <th>Data item name</th><th>Value</th><th>IN/OUT</th><th>Remark</th></tr> <tr> <td>Year</td><td>Christian era</td><td>IN</td><td>Christian year in the supported calendars</td></tr> <tr> <td>Month</td><td>1 to 12</td><td>IN</td><td></td></tr> <tr> <td>Day</td><td>1 to 31</td><td>IN</td><td></td></tr> <tr> <td rowspan="7">Day of the week</td><td>ELIB_SCH_SUN (Sunday)</td><td rowspan="7">OUT</td><td rowspan="7">The day of the week calculated and set.</td></tr> <tr> <td>ELIB_SCH_MON (Monday)</td></tr> <tr> <td>ELIB_SCH_TUE (Tuesday)</td></tr> <tr> <td>ELIB_SCH_WED (Wednesday)</td></tr> <tr> <td>ELIB_SCH_THU (Thursday)</td></tr> <tr> <td>ELIB_SCH_FRI (Friday)</td></tr> <tr> <td>ELIB_SCH_SAT (Saturday)</td></tr> <tr> <td>Hour</td><td></td><td>IN</td><td>Unused</td></tr> <tr> <td>Minute</td><td></td><td>IN</td><td>Unused</td></tr> <tr> <td>Second</td><td></td><td>IN</td><td>Unused</td></tr> </table>			Data item name	Value	IN/OUT	Remark	Year	Christian era	IN	Christian year in the supported calendars	Month	1 to 12	IN		Day	1 to 31	IN		Day of the week	ELIB_SCH_SUN (Sunday)	OUT	The day of the week calculated and set.	ELIB_SCH_MON (Monday)	ELIB_SCH_TUE (Tuesday)	ELIB_SCH_WED (Wednesday)	ELIB_SCH_THU (Thursday)	ELIB_SCH_FRI (Friday)	ELIB_SCH_SAT (Saturday)	Hour		IN	Unused	Minute		IN	Unused	Second		IN	Unused
Data item name	Value	IN/OUT	Remark																																						
Year	Christian era	IN	Christian year in the supported calendars																																						
Month	1 to 12	IN																																							
Day	1 to 31	IN																																							
Day of the week	ELIB_SCH_SUN (Sunday)	OUT	The day of the week calculated and set.																																						
	ELIB_SCH_MON (Monday)																																								
	ELIB_SCH_TUE (Tuesday)																																								
	ELIB_SCH_WED (Wednesday)																																								
	ELIB_SCH_THU (Thursday)																																								
	ELIB_SCH_FRI (Friday)																																								
	ELIB_SCH_SAT (Saturday)																																								
Hour		IN	Unused																																						
Minute		IN	Unused																																						
Second		IN	Unused																																						

Include file	srv_sch.h		
Prototype	int Elib_SCH_Week_Read (Ap_ID, Datetime);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Datetime	_ELIB_SCH_CAL_DATA *	I/O	Target date
Return value	Type	I/O	Description
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.12 Read valid date

Classification	Schedule Service supporting function																										
Description	Read valid date	Function name	Elib_SCH_DateRange_Read																								
Functional overview	<div><ul style="list-style-type: none">- This function reads a specified valid date.- This function sets the read valid date to the date and time data structure represented by the date data area.</div> <div>Date data area (date and time data structure)</div> <table><tr><td>Data item name</td><td>Value</td><td>Remark</td></tr><tr><td>Year</td><td>Christian era</td><td>Christian year in the supported calendars</td></tr><tr><td>Month</td><td>1 to 12</td><td></td></tr><tr><td>Day</td><td>1 to 31</td><td></td></tr><tr><td>Day of the week</td><td></td><td>Unused</td></tr><tr><td>Hour</td><td></td><td>Unused</td></tr><tr><td>Minute</td><td></td><td>Unused</td></tr><tr><td>Second</td><td></td><td>Unused</td></tr></table>			Data item name	Value	Remark	Year	Christian era	Christian year in the supported calendars	Month	1 to 12		Day	1 to 31		Day of the week		Unused	Hour		Unused	Minute		Unused	Second		Unused
Data item name				Value	Remark																						
Year				Christian era	Christian year in the supported calendars																						
Month				1 to 12																							
Day				1 to 31																							
Day of the week					Unused																						
Hour					Unused																						
Minute					Unused																						
Second					Unused																						
Include file				srv_sch.h																							
Prototype	int Elib_SCH_DateRange_Read(Ap_ID, Mode, Datetime);																										
Argument	Type	I/O	Description																								
Ap_ID	unsigned int	I	Application ID																								
Mode	int	I	Mode Start date and time the calendar can support: ELIB_SCH_MINDAY End date and time the calendar can support: ELIB_SCH_MAXDAY																								
Datetime	_ELIB_SCH_CAL_DATA *	O	Date data area																								
Return value	Type	I/O	Description																								
ret	int	O	Successful : ELIB_SCH_OK																								

			Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.13 Register schedule-related setting status

Classification	Schedule Service supporting function																			
Description	Register schedule-related setting status	Function name	Elib_SCH_StsSet																	
Functional overview	<div><ul style="list-style-type: none">- This function updates the setting status of calendar view switch, wake-up alarm, or alarming method.- A function name (Mode) and a status (Sts) are passed as parameters in this function. The specified function (mode) will be updated with the passed status The mode (function number) is associated with the setting status as follows:</div> <table><tr><td>Mode (function number)</td><td>Setting status</td><td>Remark</td></tr><tr><td rowspan="2">ELIB_SCH_DSP_CALEND (calendar view switch)</td><td>ELIB_SCH_STS_DSPCAL_1W (weekly view)</td><td rowspan="2"></td></tr><tr><td>ELIB_SCH_STS_DSPCAL_1M (monthly view)</td></tr><tr><td rowspan="4">ELIB_SCH_WAKE_UP (wake-up alarm)</td><td>ELIB_SCH_STS_WKALM1_ON (wake-up setting 1 enabled)</td><td rowspan="4"></td></tr><tr><td>ELIB_SCH_STS_WKALM2_ON (wake-up setting 2 enabled)</td></tr><tr><td>ELIB_SCH_STS_WKALM3_ON (wake-up setting 3 enabled)</td></tr><tr><td>ELIB_SCH_STS_OFF (disabled)</td></tr><tr><td rowspan="2">ELIB_SCH_ALM_PRI (alarming method)</td><td>ELIB_SCH_OPE_PRI (operation priority)</td><td rowspan="2"></td></tr><tr><td>ELIB_SCH_NOTICE_PRI (notification priority)</td></tr></table>			Mode (function number)	Setting status	Remark	ELIB_SCH_DSP_CALEND (calendar view switch)	ELIB_SCH_STS_DSPCAL_1W (weekly view)		ELIB_SCH_STS_DSPCAL_1M (monthly view)	ELIB_SCH_WAKE_UP (wake-up alarm)	ELIB_SCH_STS_WKALM1_ON (wake-up setting 1 enabled)		ELIB_SCH_STS_WKALM2_ON (wake-up setting 2 enabled)	ELIB_SCH_STS_WKALM3_ON (wake-up setting 3 enabled)	ELIB_SCH_STS_OFF (disabled)	ELIB_SCH_ALM_PRI (alarming method)	ELIB_SCH_OPE_PRI (operation priority)		ELIB_SCH_NOTICE_PRI (notification priority)
Mode (function number)	Setting status	Remark																		
ELIB_SCH_DSP_CALEND (calendar view switch)	ELIB_SCH_STS_DSPCAL_1W (weekly view)																			
	ELIB_SCH_STS_DSPCAL_1M (monthly view)																			
ELIB_SCH_WAKE_UP (wake-up alarm)	ELIB_SCH_STS_WKALM1_ON (wake-up setting 1 enabled)																			
	ELIB_SCH_STS_WKALM2_ON (wake-up setting 2 enabled)																			
	ELIB_SCH_STS_WKALM3_ON (wake-up setting 3 enabled)																			
	ELIB_SCH_STS_OFF (disabled)																			
ELIB_SCH_ALM_PRI (alarming method)	ELIB_SCH_OPE_PRI (operation priority)																			
	ELIB_SCH_NOTICE_PRI (notification priority)																			
Include file	srv_sch.h																			
Prototype	int Elib_SCH_StsSet(Ap_ID, Mode, Sts) ;																			
Argument	Type	I/O	Description																	
Ap_ID	unsigned int	I	Application ID																	
Mode	int	I	Mode (function number) Calendar view switch : ELIB_SCH_DSP_CALEND Wake-up alarm : ELIB_SCH_WAKE_UP Alarming method : ELIB_SCH_ALM_PRI																	
Sts	int	I	Setting status																	

			Weekly view : ELIB_SCH_STS_DSPCAL_1W Monthly view : ELIB_SCH_STS_DSPCAL_1M Wake-up setting 1 enabled : ELIB_SCH_STS_WKALM1_ON Wake-up setting 2 enabled : ELIB_SCH_STS_WKALM2_ON Wake-up setting 3 enabled : ELIB_SCH_STS_WKALM3_ON Disabled : ELIB_SCH_STS_OFF Operation priority : ELIB_SCH_OPE_PRI Notification priority : ELIB_SCH_NOTICE_PRI
Return value	Type	I/O	Description
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure: Create an MSB object by using MsbObjectCreate in respective applications.</p> <p>Note: Linux PF allows only one wake-up alarm to be "enabled (ON)." Thus, calling this function with a setting status "enabled (ON)" disables ("turns off") another alarm which has been "enabled (ON)."</p>		

5.14 Read schedule-related setting status

Classification	Schedule Service supporting function																			
Description	Read schedule-related setting status	Function name	Elib_SCH_StsRead																	
Functional overview	<div><div></div><div><ul style="list-style-type: none">- This function reads the setting status of calendar view switch, wake-up alarm, or alarming method.- This function returns the setting status of the specified function (passed in parameters). The mode is associated with the return values as follows:</div></div> <table><tr><td>Mode (function number)</td><td>Return value</td><td>Remark</td></tr><tr><td rowspan="2">ELIB_SCH_DSP_CALEND (calendar view switch)</td><td>ELIB_SCH_STS_DSPCAL_1W (weekly view)</td><td rowspan="2"></td></tr><tr><td>ELIB_SCH_STS_DSPCAL_1M (monthly view)</td></tr><tr><td rowspan="4">ELIB_SCH_WAKE_UP (wake-up alarm)</td><td>ELIB_SCH_STS_WKALM1_ON (wake-up setting 1 enabled)</td><td rowspan="4"></td></tr><tr><td>ELIB_SCH_STS_WKALM2_ON (wake-up setting 2 enabled)</td></tr><tr><td>ELIB_SCH_STS_WKALM3_ON (wake-up setting 3 enabled)</td></tr><tr><td>ELIB_SCH_STS_OFF (disabled)</td></tr><tr><td rowspan="2">ELIB_SCH_ALM_PRI (alarming method)</td><td>ELIB_SCH_OPE_PRI (operation priority)</td><td rowspan="2"></td></tr><tr><td>ELIB_SCH_NOTICE_PRI (notification priority)</td></tr></table>			Mode (function number)	Return value	Remark	ELIB_SCH_DSP_CALEND (calendar view switch)	ELIB_SCH_STS_DSPCAL_1W (weekly view)		ELIB_SCH_STS_DSPCAL_1M (monthly view)	ELIB_SCH_WAKE_UP (wake-up alarm)	ELIB_SCH_STS_WKALM1_ON (wake-up setting 1 enabled)		ELIB_SCH_STS_WKALM2_ON (wake-up setting 2 enabled)	ELIB_SCH_STS_WKALM3_ON (wake-up setting 3 enabled)	ELIB_SCH_STS_OFF (disabled)	ELIB_SCH_ALM_PRI (alarming method)	ELIB_SCH_OPE_PRI (operation priority)		ELIB_SCH_NOTICE_PRI (notification priority)
Mode (function number)	Return value	Remark																		
ELIB_SCH_DSP_CALEND (calendar view switch)	ELIB_SCH_STS_DSPCAL_1W (weekly view)																			
	ELIB_SCH_STS_DSPCAL_1M (monthly view)																			
ELIB_SCH_WAKE_UP (wake-up alarm)	ELIB_SCH_STS_WKALM1_ON (wake-up setting 1 enabled)																			
	ELIB_SCH_STS_WKALM2_ON (wake-up setting 2 enabled)																			
	ELIB_SCH_STS_WKALM3_ON (wake-up setting 3 enabled)																			
	ELIB_SCH_STS_OFF (disabled)																			
ELIB_SCH_ALM_PRI (alarming method)	ELIB_SCH_OPE_PRI (operation priority)																			
	ELIB_SCH_NOTICE_PRI (notification priority)																			
Include file	srv_sch.h																			
Prototype	int Elib_SCH_StsRead(Ap_ID, Mode) ;																			
Argument	Type	I/O	Description																	
Ap_ID	unsigned int	I	Application ID																	
Mode	int	I	Mode (function number) Calendar view switch : ELIB_SCH_DSP_CALEND Wake-up alarm : ELIB_SCH_WAKE_UP Alarming method : ELIB_SCH_ALM_PRI																	
Return value	Type	I/O	Description																	
Ret	int	O	Setting status Weekly view :																	

			ELIB_SCH_STS_DSPCAL_1W Monthly view : ELIB_SCH_STS_DSPCAL_1M Wake-up setting 1 enabled : ELIB_SCH_STS_WKALM1_ON Wake-up setting 2 enabled : ELIB_SCH_STS_WKALM2_ON Wake-up setting 3 enabled : ELIB_SCH_STS_WKALM3_ON Disabled : ELIB_SCH_STS_OFF Operation priority : ELIB_SCH_OPE_PRI Notification priority : ELIB_SCH_NOTICE_PRI Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	Necessary procedure: Create a MSB object by using MsbObjectCreate in respective applications.		

5.15 posting alarm notifications

Classification	Schedule Service supporting function		
Description	Start posting alarm notifications	Function name	Elib_SCH_AlmStart
Functional overview	<ul style="list-style-type: none"> - Applications use this function when it gets ready to receive alarm notifications. - Calling this function starts posting alarm notifications. However, alarm events generated at the system startup are (ON cause alarms) posted after all applications finish their startup sequences. - Call this function when the power is turned on. - If the time has not been set or the time has reached the end of the supported scope, no alarm is generated. - For the types of alarm notifications, see "Start requesting Schedule Service event notifications." 		
Include file	srv_sch.h		
Prototype	int Elib_SCH_AlmStart(Ap_ID) ;		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Return value	Type	I/O	Description
ret	Int	O	ELIB_SCH_OK : Successful ELIB_SCH_NG : Unsuccessful ELIB_SCH_PARAM_NG : Incorrect parameter
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.16 Register user's icon

Classification	Schedule Service supporting function		
Description	Register user's icon	Function name	Elib_SCH_UserIcon_Set
Functional overview	<p>- This function registers an original image for a schedule icon.</p> <p>- The schedule icon storage numbers that could be registered are as follows:</p> <pre> ELIB_SCH_ORG_ICON1 /* Original icon 1 */ ELIB_SCH_ORG_ICON2 /* Original icon 2 */ ELIB_SCH_ORG_ICON3 /* Original icon 3 */ ELIB_SCH_ORG_ICON4 /* Original icon 4 */ ELIB_SCH_ORG_ICON5 /* Original icon 5 */ </pre> <p>- The usable values in the 3rd argument (folder type) are as follows</p> <pre> ELIB_SCH_FOL_INBOX /* INBOX folder */ ELIB_SCH_FOL_PRIINSTALL /* Pre-installed folder */ ELIB_SCH_FOL_CAM /* Camera folder */ ELIB_SCH_FOL_ORGANIME /* User-defined animation folder */ ELIB_SCH_FOL_USER01 /* User-created folder 01 */ : ELIB_SCH_FOL_USER20 /* User-created folder 20 */ </pre> <p>(Exception)</p> <p>To delete a user's icon, set the following value to the 3rd argument (folder type):</p> <pre> ELIB_SCH_FOL_NOSET /* No folder (unset) */ </pre> <p>- The data to be registered must be the number of an original image. Set the image ID of an original image provided by the resource management library (image data storage number) to the 4th argument (data to be registered).</p> <p>(Exception)</p> <p>When ELIB_SCH_FOL_NOSET (No folder (unset)) is specified in the 3rd argument (folder type), the 4th argument (data to be registered) must be unused.</p>		

<Note>

- When an user's icon is deleted with "ELIB_SCH_FOL_NOSET" specified for the folder type and if the original icon is being used for already registered schedule data, the icon of the schedule data is changed to the default icon (ELIB_SCH_ICON_DATA1).

Include file	srv_sch.h		
Prototype	Int Elib_SCH_UserIcon_Set(Ap_ID, No, FolNo, Data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
No	int	I	Schedule icon storage number
FolNo	int *	I	Folder type INBOX folder : ELIB_SCH_FOL_INBOX Pre-installed folder : ELIB_SCH_FOL_PRIINSTALL Camera folder : ELIB_SCH_FOL_CAM User-defined animation folder : ELIB_SCH_FOL_ORGANIME User-created folder 01 : ELIB_SCH_FOL_USER01 : User-created folder 20 : ELIB_SCH_FOL_USER20 No folder (unset) : ELIB_SCH_FOL_NOSET
Data	unsigned short *	I	Registered data storage field
Return value	Type	I/O	Description
ret	int	O	ELIB_SCH_OK : Successful ELIB_SCH_NG : Unsuccessful ELIB_SCH_PARAM_NG : Incorrect parameter
Remark	<p>Necessary procedure:</p> <p>Create an MSB object by using MsbObjectCreate in respective applications.</p>		

5.17 Read user's icon

Classification	Schedule Service supporting function		
Description	Read user's icon	Function name	Elib_SCH_UserIcon_Read
Functional overview	<ul style="list-style-type: none"> - This function reads the folder type and image ID of an original image specified for a schedule icon. The readable icon numbers are 1 to 5. - The readable schedule icon storage numbers are as follows: <ul style="list-style-type: none"> ELIB_SCH_ORG_ICON1 /* Original icon 1 */ ELIB_SCH_ORG_ICON2 /* Original icon 2 */ ELIB_SCH_ORG_ICON3 /* Original icon 3 */ ELIB_SCH_ORG_ICON4 /* Original icon 4 */ ELIB_SCH_ORG_ICON5 /* Original icon 5 */ - Any of the following values is output to the 3rd argument (folder type): <ul style="list-style-type: none"> ELIB_SCH_FOL_INBOX /* INBOX folder */ ELIB_SCH_FOL_PRIINSTALL /* Pre-installed folder */ ELIB_SCH_FOL_CAM /* Camera folder */ ELIB_SCH_FOL_ORGANIME /* User-defined animation folder */ ELIB_SCH_FOL_USER01 /* User-created folder 01 */ : ELIB_SCH_FOL_USER20 /* User-created folder 20 */ <p>(Exception)</p> <p>If the schedule icon has been registered (unset), the following value is output to the 3rd argument (folder type):</p> <ul style="list-style-type: none"> ELIB_SCH_FOL_NOSET /* No folder (unset) */ <ul style="list-style-type: none"> - The data to be referenced must be the image ID of an original image. <p>To the 4th argument (registered data storage field), the image ID of an original image provided by the resource management library (image data storage number) is output.</p> <p>(Exception)</p> <p>If no user icon has been set to the specified schedule icon storage number, that is, if ELIB_SCH_FOL_NOSET (No folder (unset)) is output to the 3rd argument (folder type), the 4th argument (data to be registered) must be unused.</p>		

Include file	srv_sch.h		
Prototype	int Elib_SCH_UserIcon_Read(Ap_ID, No, FolNo, Data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
No	int	I	Schedule icon storage number
FolNo	int *	O	Folder type INBOX folder : ELIB_SCH_FOL_INBOX Pre-installed folder : ELIB_SCH_FOL_PRIINSTALL Camera folder : ELIB_SCH_FOL_CAM User-defined animation folder : ELIB_SCH_FOL_ORGANIME User-created folder 01 : ELIB_SCH_FOL_USER01 User-created folder 20 : ELIB_SCH_FOL_USER20 No folder (unset) : ELIB_SCH_FOL_NOSET
Data	unsigned short *	O	Registered data storage field
Return value	Type	I/O	Description
ret	int	O	ELIB_SCH_OK : Successful ELIB_SCH_NG : Unsuccessful ELIB_SCH_PARAM_NG : Incorrect parameter
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.18 Check for original data in use

Classification	Schedule Service supporting function		
Description	Check for original data in use	Function name	Elib_SCH_OrgData_Search
Functional overview	<ul style="list-style-type: none"> - This function checks whether a specified original data icon is being used in regular schedules. - When the specified original data icon is used in one or more regular schedules, Used (ELIB_SCH_USE) is returned. - When the specified original data icon is not used in any regular schedule, Unused (ELIB_SCH_NOTUSE) is returned. - The specifiable original data icons are as follows: <ul style="list-style-type: none"> ELIB_SCH_ORG_ICON1 /* Original icon 1 */ ELIB_SCH_ORG_ICON2 /* Original icon 2 */ ELIB_SCH_ORG_ICON3 /* Original icon 3 */ ELIB_SCH_ORG_ICON4 /* Original icon 4 */ ELIB_SCH_ORG_ICON5 /* Original icon 5 */ - This function makes search not only data registered for a specified application ID but also all of already registered data. 		
Include file	srv_sch.h		
Prototype	int Elib_SCH_OrgData_Search(Ap_ID, Data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Data	short	I	Original data
Return value	Type	I/O	Description
ret	int	O	ELIB_SCH_USE : Used ELIB_SCH_NOTUSE : Unused ELIB_SCH_NG : Unsuccessful ELIB_SCH_PARAM_NG : Incorrect parameter
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.19 Delete multiple schedules

Classification	Schedule Service supporting function		
Description	Delete multiple schedules	Function name	Elib_SCH_Plural_Data_Delete
Functional overview	<ul style="list-style-type: none"> - This function deletes multiple schedules. - Specify the number of a schedule to be deleted by using the bits for the schedule number array. - In the 2nd argument, specify an int-type pointer to the array which has the number of elements indicated by ELIB_SCH_MAX_INFO. <p><Example></p> <pre>int SelectInfo[ELIB_SCH_MAX_INFO] ;</pre> <p>SelectInfo[0]</p> <p>When BIT0 - ON, the schedule number 1 is selected and set as a schedule to be deleted.</p> <p>When BIT1 - ON, the schedule number 2 is selected and set as a schedule to be deleted.</p> <p>:</p> <p>When BIT31 - ON, the schedule number 32 is selected and set as a schedule to be deleted.</p> <p>SelectInfo[1]</p> <p>When BIT0 - ON, the schedule number 33 is selected and set as a schedule to be deleted.</p> <p>:</p> <p>Therefore, this function allows up to SelectInfo[ELIB_SCH_MAX_INFO - 1] to be set.</p> <ul style="list-style-type: none"> - For any bits left unused up to SelectInfo[ELIB_SCH_MAX_INFO - 1], turn off the bits. - This function sends a completion event as soon as returns "finished." (For details about events, see "A-1 Start requesting Schedule Service event notifications.") 		
Include file	srv_sch.h		
Prototype	int Elib_SCH_Plural_Data_Delete(Ap_ID, SelectInfo_p);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
SelectInfo_p	int *	I/O	Array of numbers of schedules to be deleted
Return value	Type	I/O	Description

ret	int	O	ELIB_SCH_OK : Successful ELIB_SCH_NG : Unsuccessful ELIB_SCH_PARAM_NG : Incorrect parameter
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.20 Set system date/time

Classification	Calendar feature supporting function		
Description	Set system date/time	Function name	Elib_SCH_Calendar_Set
Functional overview	<div><div><div><div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></</div></div></div></div></div></div>		

Return value	Type		
ret	int	0	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter: ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.21 Read system date/time

Classification	Calendar feature supporting function		
Description	Read system date/time	Function name	Elib_SCH_Calendar_Read
Functional overview			
<ul style="list-style-type: none"> - This function reads the system date and time. - This function sets the system date and time into the date and time data structure represented by the specified date/time readout area. (For details about the structure, see "4. List of Structures.") - Specified date/time readout area (date and time data structure) 			
Data item name	Value	Remark	
Year	Christian era	Christian year in the supported calendars	
Month	1 to 12		
Day	1 to 31		
Day of the week	0 to 6 (corresponding to Sun to Sat)		
Hour	0 to 23		
Minute	0 to 59		
Second	0 to 59		
Include file	Srv_sch.h		
Prototype	int Elib_SCH_Calendar_Read(Ap_ID, datetime) ;		
Argument	Type	I/O	Description
Ap_ID	Unsigned int	I	Application ID
datetime	_ELIB_SCH_CAL_DATA *	O	Specified date/time readout area
Return value	Type	-	
ret	int	O	Successful : ELIB_SCH_OK Unsuccessfu : ELIB_SCH_NG Date and time unset : ELIB_SCH_CAL_NOTSET Incorrect parameter :

			ELIB_SCH_PARAM_NG
Remark			
<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>			

5.22 Convert specified date and time to seconds

Classification	Calendar feature supporting function		
Description	Convert specified date and time to seconds	Function name	Elib_SCH_DateTimeToSec
Functional overview	<div><div><div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div></div></div></div>		

		Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure:</p> <p>Create an MSB object by using MsbObjectCreate in respective applications.</p>	

		Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>	

			ELIB_SCH_CAL_HM
DateTimePrev	_ELIB_SCH_CALPREV_DATA *	I	Date and time (prior notice) area
Return value	Type		
ret	Int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.25 Convert seconds to date and time (GMT time zone)

Classification	Calendar feature supporting function																										
Description	Convert seconds to date and time (GMT time zone)	Function name	Elib_SCH_SecToDateTime_Atn																								
Functional overview	<div><div><div></div><div></div><div></div></div><div><ul style="list-style-type: none">- This function converts seconds from the reference date and time the calendar can support to the form of year, month, day, hour, minute, and second.- This function sets the converted date and time into the date and time data structure represented by the converted date and time data area.- Converted date and time data area (date and time data structure)</div></div> <table><tr><td>Data item name</td><td>Value</td><td>Remark</td></tr><tr><td>Year</td><td>Christian era</td><td></td></tr><tr><td>Month</td><td>1 to 12</td><td></td></tr><tr><td>Day</td><td>1 to 31</td><td></td></tr><tr><td>Day of the week</td><td>0 to 6 (corresponding to Sun to Sat)</td><td></td></tr><tr><td>Hour</td><td>0 to 23</td><td></td></tr><tr><td>Minute</td><td>0 to 59</td><td></td></tr><tr><td>Second</td><td>0 to 59</td><td></td></tr></table>			Data item name	Value	Remark	Year	Christian era		Month	1 to 12		Day	1 to 31		Day of the week	0 to 6 (corresponding to Sun to Sat)		Hour	0 to 23		Minute	0 to 59		Second	0 to 59	
Data item name	Value	Remark																									
Year	Christian era																										
Month	1 to 12																										
Day	1 to 31																										
Day of the week	0 to 6 (corresponding to Sun to Sat)																										
Hour	0 to 23																										
Minute	0 to 59																										
Second	0 to 59																										
Include file	srv_sch.h																										
Prototype	int Elib_SCH_SecToDateTime_Atn (Ap_ID, sec, datetime) ;																										
Argument	Type	I/O	Description																								
Ap_ID	unsigned int	I	Application ID																								
sec	unsigned long	I	Specified seconds to be converted																								
datetime	_ELIB_SCH_CAL_DATA *	O	Converted date and time data area																								
Return value	Type	I/O	Description																								
ret	int	O	Successful : ELIB_SCH_OK																								

		Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>	

5.26 Register wake-up alarm

Classification	Wake-up alarm supporting function		
Description	Register alarm	wake-up	Function name Elib_SCH_Ex_Set
Functional overview	<ul style="list-style-type: none"> - This function registers a wake-up alarm. - Contents set into the wake-up data structure represented by the wake-up alarm area is registered as a wake-up alarm. - The number of wake-up alarm settings allowed to be registered must be 3, and only one of the 3 settings can be enabled. - Any data registered with "one-off" set for the schedule type are cancelled if the time set for it has passed by. - If there are already registered data, this function overwrites the data. - For the alarm sound setting, specify a sound ID associated by Sound System Service. - Calling this function after the wake-up alarm function has been specified allows to use the sound recorded by the voice recording feature (voice recording data). - When voice recording data are present, they are added to the alarm sound list created by Sound System Service. - If the original melody or voice recording data specified for the ringing sound of the wake-up alarm has been deleted, the ringing sound is reset to the default. - To play a sound of voice recording data, use the ring interface of Recording/Playback Service. To play another alarm sound, use the ring interface of Sound System Service in alarm ringing on alarm sound selection/event occurrence. - To post event notifications, "enable" the setting status and register valid data. This function does not update the setting status. To update the "enabled/disabled" setting status, call "A-17 Register schedule-related setting status" before calling this function to update the setting status. 		

- To receive notifications about an event specified by this function, the application needs to make a call to the "Start posting alarm notifications" function when it starts up to inform that the application is ready to hand the event.
- Regular alarm (regular schedule) event notifications are posted even if the power is turned on due to a power-on factor alarm, thus the application must be designed to ignore unnecessary notifications.
- For the weekly schedule, see "Register schedule event."

Include file	srv_sch.h		
Prototype	int Elib_SCH_Ex_Set(Ap_ID, Sch_Data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Sch_Data	_ELIB_SCH_EX_DATA *	I	Wake-up alarm area
Return value	Type	I/O	Description
ret	Int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark			

Necessary procedure:

Create a MSB object by using MsbObjectCreate in respective applications.

5.27 Delete wake-up alarm

Classification	Wake-up alarm supporting function		
Description	Delete wake-up alarm	Function name	Elib_SCH_Ex_Reset
Functional overview	<ul style="list-style-type: none"> - This function deletes data of a wake-up alarm registered by using the "C-1 Register wake-up alarm" function. - This function does not update the setting status. To update the "enabled/disabled" setting status, call "A-17 Register schedule-related setting status" before calling this function to update the setting status. 		
Include file	srv_sch.h		
Prototype	Int Elib_SCH_Ex_Reset(Ap_ID, Mode, Wakeup_No);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Mode	int	I	Type Wake-up alarm: ELIB_SCH_WAKE_UP
Wakeup_No	int	I	Number of the wake-up alarm to be deleted All wake-up alarms :ELIB_SCH_WAKEUP_ALL Wake-up number 1 :ELIB_SCH_WAKEUP_NO1 Wake-up number 2 :ELIB_SCH_WAKEUP_NO2 Wake-up number 3 :ELIB_SCH_WAKEUP_NO3
Return value	Type	I/O	Description
Ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure:</p> <p>Create an MSB object by using MsbObjectCreate in respective applications.</p>		

5.28 Read wake-up alarm

Classification	Wake-up alarm supporting function		
Description	Read wake-up alarm	Function name	Elib_SCH_Ex_Read
Functional overview	<ul style="list-style-type: none"> - This function reads data setting information registered by using the "C-1 Register wake-up alarm" function. - This function sets readout conditions into the wake-up data structure represented by the wake-up alarm readout area. The result is put into the said structure. <p>For information about time, the "hour and minute, and alarm sound" are to be set. (The year, month, day, and second are unused. The day of the week is calculated and set according to the year, month, and day.)</p> <ul style="list-style-type: none"> - The day of the week can be specified for the wake-up alarm, thus the function output area for wake-up data (year, month, day, hour, minute, second) are set as follows: <ul style="list-style-type: none"> (1) When the alarm can be enabled for the next time, the year, month, day, hour, and minute of the date on which the alarm is output next time to the output date. (2) When the alarm cannot be enabled for the next time, the start date and time the calendar can support and the specified hour and minute are set to the output date. 		
Include file	srv_sch.h		
Prototype	int Elib_SCH_Ex_Read(Ap_ID, Sch_Data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Sch_Data	_ELIB_SCH_EX_DATA *	I/O	Wake-up alarm readout area
Return value	Type	I/O	Description
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure:</p>		

Create a MSB object by using `MsbObjectCreate` in respective applications.

Notes:

1. For any wake-up alarm not to be repeated, re-setting the battery after the specified time for a wake-up alarm invalidates the wake-up alarm to be output next time.
2. If the wake-up data have not been specified or has been set up with invalid data, a return is made with 0xFF set into the `_ELIB_SCH_EX_DATA` structure members Hour and Min.

5.29 Register one ToDo data item

Classification	ToDo feature supporting function		
Description	Register one ToDo data item	Function name	Elib_ToDo_Data_Set
Functional overview	<ul style="list-style-type: none"> - Applications call this function to register ToDo data. - A maximum of 100 ToDo data could be registered. - This function sets the data to be registered into the ToDo data structure represented by the ToDo data to be registered. The result is put into the said structure. - For expired member data, this function is to be called by the application with an unset value (unfixed value), but the said member data are re-calculated and set within Schedule Service. - On successful registration, the "day of the week and expiration flag" of the ToDo data structure are automatically set by Schedule Service. In addition, for ToDo data newly registered, the "ToDo number" is also automatically set. - For any data registered without a specified date, the application needs to register it setting zeros to the year, month, and day. - To receive notifications about an event specified by this interface (function), the application needs to make a call to the "A-19 Start posting alarm notifications" function when it starts up to inform that the application is ready to hand the event. - If the alarm type is set to "Alarm turned off," the application is notified of the event. - The time for prior notice is output as it is regardless of the alarm type. <p><Alarm sound settings></p> <ul style="list-style-type: none"> - Sound ID Specify a sound ID associated by Sound System Service. - Alarm sound allowed to set <ul style="list-style-type: none"> - Alarm sounds for the schedule features provided by Sound System Service - Sound recorded by using the voice recording feature (voice recording data) - If the original melody deleted 		

- If the original melody specified for the alarm sound is deleted, the alarm sound setting is restored to the default setting.
- Voice recording data
When there is voice recording data, the voice recording data are added to the alarm sound list created by Sound System Service.
- If the voice recording data deleted
If the voice recording data specified for the alarm sound is deleted, the alarm sound setting is restored to the default setting.
- Alarm ringing on alarm sound selection/event occurrence
To play a sound of voice recording data, use the ring interface of Recording/Playback Service. To play another alarm sound, use the ring interface of Sound System Service.

Include file	srv_sch.h		
Prototype	int Elib_ToDo_Data_Set (Ap_ID, ToDo_Data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
ToDo_Data	_ELIB_TODO_DATA *	I/O	ToDo data to be registered
Return value	Type	I/O	Description
ret	Int	O	Successful : ELIB_TODO_OK Unsuccessful : ELIB_TODO_NG Incorrect parameter : ELIB_TODO_PARAM_NG Unregistered data : ELIB_TODO_NODATA Max registered Events reached : ELIB_TODO_MAXDATA
Remark			

Necessary procedure:

Create a MSB object by using `MsbObjectCreate` in respective applications.

5.30 Delete one ToDo data item

Classification	ToDo feature supporting function		
Description	Delete one ToDo data item	Function name	Elib_ToDo_Specified_Data_Delete
Functional overview	<ul style="list-style-type: none"> - This function deletes data corresponding to a specified ToDo number. - If this function cannot find the ToDo data to be deleted, it returns with "Unregistered." 		
Include file	srv_sch.h		
Prototype	int Elib_ToDo_Specified_Data_Delete(Ap_ID, ToDo_No);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
ToDo_No	Int	I	ToDo number to be deleted (1 to 100)
Return value	Type	I/O	Description
ret	Int	O	Successful : ELIB_TODO_OK Unsuccessful : ELIB_TODO_NG Incorrect parameter : ELIB_TODO_PARAM_NG Unregistered : ELIB_TODO_NODATA
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.31 Delete ToDo data matching specified conditions

Classification	ToDo feature supporting function		
Description	Delete ToDo data matching specified conditions	Function name	Elib_ToDo_Specified_Kind_Delete
Functional overview	<p>- This function deletes ToDo data (finished only) matching specified conditions.</p>		
Include file	srv_sch.h		
Prototype	int Elib_ToDo_Specified_Kind_Delete(Ap_ID, KindOfDel);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
KindOfDel	int	I	Conditions to delete Finished: ELIB_TODO_DEL_BY_FINISHED
Return value	Type	I/O	Description
Ret	int	O	Successful : Number of data deleted Unsuccessful : ELIB_TODO_NG Incorrect parameter : ELIB_TODO_PARAM_NG
Remark	<p>Necessary procedure: Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.32 Delete all ToDo data

Classification	ToDo feature supporting function		
Description	Delete all ToDo data	Function name	Elib_ToDo_All_Data_Delete
Functional overview	<p>- This function deletes all registered ToDo data.</p>		
Include file	srv_sch.h		
Prototype	int Elib_ToDo_All_Data_Delete(Ap_ID);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Return value	Type	I/O	Description
ret	int	O	Successful : Number of data deleted Unsuccessful : ELIB_TODO_NG Incorrect parameter : ELIB_TODO_PARAM_NG
Remark	<p>Necessary procedure: Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.33 Read one ToDo data item

Classification	ToDo feature supporting function		
Description	Read one ToDo data item	Function name	Elib_ToDo_Specified_Data_Read
Functional overview	<ul style="list-style-type: none"> - This function reads data registered with a specified ToDo number. - If this function cannot find the ToDo data to be read, it returns with "Unregistered." - This function sets the readout result into the ToDo data structure represented by the ToDo data. 		
Include file	srv_sch.h		
Prototype	int Elib_ToDo_Specified_Data_Read(Ap_ID, No, ToDo_Data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
No	Int	I	ToDo number (1 to 100)
ToDo_Data	_ELIB_TODO_DATA *	O	ToDo data
Return value	Type	I/O	Description
Ret	Int	O	Successful : ELIB_TODO_OK Unsuccessful : ELIB_TODO_NG Incorrect parameter : ELIB_TODO_PARAM_NG Unregistered : ELIB_TODO_NODATA
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.34 Read one ToDo data item by category

Classification	ToDo feature supporting function		
Description	Read ToDo data by category	Function name	Elib_ToDo_Category_Read
Functional overview	<ul style="list-style-type: none"> - This function reads ToDo data in a specified category one by one. - This function sets read conditions into the ToDo category conditioning structure represented by the readout conditioning area. Data matching the conditions is set into the ToDo data structure represented by the ToDo data area. - This function lists ToDo information matching the specified conditions at the initial call to it (made with 0 set to the listing number argument). In the second or later call, specifying the position of ToDo in the listing number argument allows the ToDo data to be read. <p>[Sorting conditions]</p> <ul style="list-style-type: none"> - When any of the conditions below is specified, data are listed with the sorting condition of "by registration" included: "planning" "accepted" "requesting" "provisional" "confirmation" "rejected" "entrusted" "finished" - When the condition specified is "by date," the following sort conditions are used for listing: <ul style="list-style-type: none"> (1) As a whole, data are sorted by date from the earliest one and data without a date specified are listed by registration in the end of the list. (2) Data with the same date specified are sorted by priority (high → low → none). (3) Data with the same date and priority specified are sorted by registration. (4) For data "without a date specified," the same-date rule above-mentioned is applied. - When the condition specified is "by finish date," the following sort conditions are used for listing: <ul style="list-style-type: none"> (1) As a whole, data are sorted by finish date from the earliest one and data without a finish date specified are listed by registration in the end of the list. (2) Data with the same finish date specified are sorted by priority (high → low → none). (3) Data with the same finish date and priority specified are sorted by registration. (4) For data "without a date specified," the same-date rule above-mentioned is applied. 		
Include file	srv_sch.h		

Prototype	int Elib_ToDo_Category_Read(Ap_ID, List_No, In, Out, ToDo_Data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
List_No	int	I	Listing number
In	_ELIB_TODO_CATG _COND *	I	Readout conditioning area
Out	_ELIB_TODO_CATG _COND *	O	Readout condition result area
ToDo_Data	_ELIB_TODO_DATA *	O	ToDo data area
Return value	Type	I/O	Description
ret	int	O	Successful : Number of data matching the conditions Unsuccessful : ELIB_TODO_NG Unregistered : ELIB_TODO_NODATA Incorrect parameter : ELIB_TODO_PARAM_NG
Remark	<p>Necessary procedure:</p> <p> Create a MSB object by using MsbObjectCreate in respective applications.</p> <p>Notes:</p> <ol style="list-style-type: none"> 1. The information put in the list by the initial call comes from static areas of functions. Therefore, the caller must note that an initial call made before it reads all schedule data modifies static areas. 		

5.35 Get number of registered ToDo data items

Classification	ToDo feature supporting function		
Description	Get number of registered ToDo data	Function name	Elib_ToDo_Total_By_Category
Functional overview	<p>- This function gets the number of ToDo data items registered in a specified category. Setting "All" to the category makes this function to get the number of all registered data items.</p>		
Include file	srv_sch.h		
Prototype	int Elib_ToDo_Total_By_Category(Ap_ID, Category);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Category	int	I	Category All : ELIB_TODO_CATG_ALL None : ELIB_TODO_CATG_NONE Plan : ELIB_TODO_CATG_PLAN Personal affair : ELIB_TODO_CATG_PRIVATE Holiday : ELIB_TODO_CATG_HOLIDAY Trip : ELIB_TODO_CATG_TRAVEL Work : ELIB_TODO_CATG_WORK Meeting : ELIB_TODO_CATG_MEETING
Return value	Type	I/O	Description
ret	int	O	Successful : Number of data items registered Unsuccessful : ELIB_TODO_NG Incorrect parameter : ELIB_TODO_PARAM_NG
Remark	<p>Necessary procedure: Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.36 Set ToDo data status

Classification	ToDo feature supporting function		
Description	Set ToDo data status	Function name	Elib_ToDo_Status_Set
Functional overview	<ul style="list-style-type: none"> - This function sets the status of ToDo data with a specified ToDo number. - If this function cannot find the target ToDo data to be handled, it returns with "Unregistered." - When setting "finished" to the status, specify "date and time of finish." - On successful setting, this function sets the information about the ToDo data into the ToDo data structure represented by the ToDo data area. 		
Include file	srv_sch.h		
Prototype	int Elib_ToDo_Status_Set(Ap_ID, ToDo_No, Sts, ToDo_Data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
ToDo_No	int	I	ToDo number (1 to 100)
Sts	int	I	Status Planning : ELIB_TODO_PLAN Accepted : ELIB_TODO_CONSENT Requesting : ELIB_TODO_REQUEST Provisional : ELIB_TODO_TENTATIVE Confirmation : ELIB_TODO_CONFIRM Rejected : ELIB_TODO_REFUSAL Entrusted : ELIB_TODO_AGENT Finished : ELIB_TODO_FINISHED
ToDo_Data	_ELIB_TODO_DATA *	I/O	ToDo data
Return value	Type	I/O	Description
ret	int	O	Successful : ELIB_TODO_OK Unsuccessful : ELIB_TODO_NG Incorrect

			parameter : ELIB_TODO_PARAM_NG Unregistered : ELIB_TODO_NODATA
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.37 Set number of ToDo data items by category and status

Classification	ToDo feature supporting function																																							
Description	Set number of ToDo data items by category and status	Function name	Elib_ToDo_Total_By_Status																																					
Functional overview																																								
<ul style="list-style-type: none">- This function gets the number of ToDo data items in a specified status and category.- This function sets conditions to acquire the number of data items into the ToDo category retrieval conditioning structure represented by the retrieval conditions. (For details about the structure, see "4. List of Structures.")- Retrieval conditions (ToDo category retrieval conditioning structure)																																								
<table><tr><th>Data item name</th><th>Value</th><th>Remark</th></tr><tr><td rowspan="8">Category</td><td>ELIB_TODO_CATG_ALL (all)</td><td></td></tr><tr><td>ELIB_TODO_CATG_NONE (none)</td><td></td></tr><tr><td>ELIB_TODO_CATG_PLAN (plan)</td><td></td></tr><tr><td>ELIB_TODO_CATG_PRIVATE (personal affair)</td><td></td></tr><tr><td>ELIB_TODO_CATG_HOLIDAY (holiday)</td><td></td></tr><tr><td>ELIB_TODO_CATG_TRAVEL (trip)</td><td></td></tr><tr><td>ELIB_TODO_CATG_WORK (work)</td><td></td></tr><tr><td>ELIB_TODO_CATG_MEETING (meeting)</td><td></td></tr><tr><td rowspan="8">Condition</td><td>ELIB_TODO_PLAN (planning)</td><td></td></tr><tr><td>ELIB_TODO_CONSENT (accepted)</td><td></td></tr><tr><td>ELIB_TODO_REQUEST (requesting)</td><td></td></tr><tr><td>ELIB_TODO_TENTATIVE (provisional)</td><td></td></tr><tr><td>ELIB_TODO_CONFIRM (confirmation)</td><td></td></tr><tr><td>ELIB_TODO_REFUSAL (rejected)</td><td></td></tr><tr><td>ELIB_TODO_AGENT (entrusted)</td><td></td></tr><tr><td>ELIB_TODO_FINISHED (finished)</td><td></td></tr></table>				Data item name	Value	Remark	Category	ELIB_TODO_CATG_ALL (all)		ELIB_TODO_CATG_NONE (none)		ELIB_TODO_CATG_PLAN (plan)		ELIB_TODO_CATG_PRIVATE (personal affair)		ELIB_TODO_CATG_HOLIDAY (holiday)		ELIB_TODO_CATG_TRAVEL (trip)		ELIB_TODO_CATG_WORK (work)		ELIB_TODO_CATG_MEETING (meeting)		Condition	ELIB_TODO_PLAN (planning)		ELIB_TODO_CONSENT (accepted)		ELIB_TODO_REQUEST (requesting)		ELIB_TODO_TENTATIVE (provisional)		ELIB_TODO_CONFIRM (confirmation)		ELIB_TODO_REFUSAL (rejected)		ELIB_TODO_AGENT (entrusted)		ELIB_TODO_FINISHED (finished)	
Data item name	Value	Remark																																						
Category	ELIB_TODO_CATG_ALL (all)																																							
	ELIB_TODO_CATG_NONE (none)																																							
	ELIB_TODO_CATG_PLAN (plan)																																							
	ELIB_TODO_CATG_PRIVATE (personal affair)																																							
	ELIB_TODO_CATG_HOLIDAY (holiday)																																							
	ELIB_TODO_CATG_TRAVEL (trip)																																							
	ELIB_TODO_CATG_WORK (work)																																							
	ELIB_TODO_CATG_MEETING (meeting)																																							
Condition	ELIB_TODO_PLAN (planning)																																							
	ELIB_TODO_CONSENT (accepted)																																							
	ELIB_TODO_REQUEST (requesting)																																							
	ELIB_TODO_TENTATIVE (provisional)																																							
	ELIB_TODO_CONFIRM (confirmation)																																							
	ELIB_TODO_REFUSAL (rejected)																																							
	ELIB_TODO_AGENT (entrusted)																																							
	ELIB_TODO_FINISHED (finished)																																							

Include file	srv_sch.h		
Prototype	int Elib_ToDo_Total_By_Status(Ap_ID, StsCatg_p);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
StsCatg_p	ELIB_TODO_CAT G_COND *	I	Retrieval conditions
Return value	Type	I/O	Description
ret	int	O	Successful : Number of matching data items Unsuccessful : ELIB_TODO_NG Incorrect parameter : ELIB_TODO_PARAM_NG
Remark	<p>Necessary procedure:</p> <p> Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.38 Search for ToDo data not alarmed

Classification	ToDo feature supporting function		
Description	Search for ToDo data not alarmed	Function name	Elib_ToDo_ActiveAlmSearch
Functional overview	<p>- This function checks for any ToDo data registered but not alarmed.</p> <p>- The calendared alarm date and time is used to search for those not alarmed.</p>		
Include file	srv_sch.h		
Prototype	Int Elib_ToDo_ActiveAlmSearch(Ap_ID);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Return value	Type	I/O	Description
ret	int	O	<p>Found (those on and after the present time) : ELIB_TODO_ALM_TODAY</p> <p>Found (those on tomorrow or later) : ELIB_TODO_ALM_AFTER</p> <p>Not found : ELIB_TODO_NODATA</p> <p>Unsuccessful : ELIB_TODO_NG</p> <p>Incorrect parameter : ELIB_TODO_PARAM_NG</p>
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p> <p>Note:</p> <p>The calendared alarm date and time includes the time for prior notice.</p>		

5.39 Delete multiple ToDo data items

Classification	ToDo feature supporting function		
Description	Delete multiple ToDo data items	Function name	Elib_ToDo_Plural_Data_Delete
Functional overview	<ul style="list-style-type: none"> - This function deletes multiple ToDo data items. - Set the ToDo numbers of data items to be deleted into the bits of the ToDo number array for deletion. - To the 2nd argument, specify an int-type pointer to the array which has the number of elements indicated by ELIB_TODO_MAX_INFO. <p>[Setting example]</p> <pre>int SelectInfo[ELIB_TODO_MAX_INFO] ;</pre> <p>SelectInfo[0]</p> <p>When BIT0 - ON, the ToDo number 1 is selected and set as data to be deleted.</p> <p>When BIT1 - ON, the ToDo number 2 is selected and set as data to be deleted.</p> <p>:</p> <p>When BIT31 - ON, the ToDo number 32 is selected and set as data to be deleted.</p> <p>SelectInfo[1]</p> <p>When BIT0 - ON, the ToDo number 33 is selected and set as data to be deleted.</p> <p>:</p> <p>Therefore, this function allows up to SelectInfo[ELIB_TODO_MAX_INFO - 1] to be set.</p> <ul style="list-style-type: none"> - For any bits left unused up to SelectInfo[ELIB_TODO_MAX_INFO - 1], turn off the bits. - This function sends a completion event as soon as returns "finished." (For details about events, see "A-1 Start requesting Schedule Service event notifications.") 		
Include file	srv_sch.h		

Prototype	int Elib_ToDo_Plural_Data_Delete(Ap_ID, SelectInfo_p);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
SelectInfo_p	int *	I/O	Array of numbers of ToDo data to be deleted
Return value	Type	I/O	Description
ret	Int	O	Successful : ELIB_TODO_OK Unsuccessful : ELIB_TODO_NG Incorrect parameter : ELIB_TODO_PARAM_NG
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.40 Register Java schedule event

Classification	Java supporting function		
Description	Register Java schedule event	Function name	Elib_SCH_Java_Set
Functional overview	<ul style="list-style-type: none"> - Applications call this interface (function) to register a Java schedule. - The number of Java schedule which can be registered must be 3. - If the number of already registered Java schedule events has reached the maximum, "Number of events registered reaching maximum" is returned then this function quits. - Set data into the Java schedule data structure, then call this interface (function). - When a former schedule event is to be overwritten but if the specified schedule number has not been registered, "Unregistered data" is returned and this function quits. - When the schedule type is "weekly schedule," the date is calculated as stated in (1) or (2) then output to the "year, month, and day" members: <ul style="list-style-type: none"> (1) When the date specified for the data is before the system date and "day of the week selection" is applied to the day of the week of the system date: <ul style="list-style-type: none"> → The system date is output. (2) Other than (1) <ul style="list-style-type: none"> → The date on which the alarm is emitted next time is calculated and the result of calculation is output. <p>(Note) The "year, month, and day" are to be output, the day of the week of the date is output to the "day of the week" member.</p> - For the date and time, set the "year, month, day, hour, minute, second, and day of the week." - When the schedule type is set to other than "isochronal schedule," 0 is always set to the "second." - When the schedule type is set to "isochronal schedule," set the reference download time to the date and time data. - To receive notifications about an event specified by this interface (function), the application needs to make a call to the "A-19 Start posting alarm notifications" function when it starts up to 		

inform that the application is ready to process the event.

(For the format of events posted from Schedule Service to service-requesting applications, see "A-1 Start requesting Schedule Service event notifications.")

- For the specified Java schedule alarm, event notifications are posted to the application whether a UIM card is used or not used.

Include file	srv_sch.h		
Prototype	Int Elib_SCH_Java_Set(Ap_ID, Entry_ID, Sch_Data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Entry_ID	Long	I	Entry ID (i-appli ID)
Sch_Data	_ELIB_SCH_JAVA_DATA *	I/O	Schedule data to be registered
Return value	Type	I/O	Description
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered data : ELIB_SCH_NODATA Max registered Events reached : ELIB_SCH_MAXDATA Weekly specified setting out of scope : ELIB_SCH_WEEKOVERFLOW Not supported year : ELIB_SCH_OVERFLOW
Remark			

Necessary procedure:

Create a MSB object by using `MsbObjectCreate` in respective applications.

5.41 Delete Java schedule event

Classification	Java supporting function		
Description	Delete Java schedule event	Function name	Elib_SCH_Java_Reset
Functional overview			
<ul style="list-style-type: none"> - This function deletes a schedule and stops event notifications of the schedule to the application. - If this function cannot find the schedule to be deleted, it returns with "Unregistered." 			
Include file	srv_sch.h		
Prototype	Int Elib_SCH_Java_Reset(Ap_ID, Schedule_No);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Schedule_No	unsigned short	I	Number of the schedule to be deleted
Return value	Type	I/O	Description
ret	Int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered data : ELIB_SCH_NODATA
Remark			
Necessary procedure: Create a MSB object by using MsbObjectCreate in respective applications.			

5.42 Delete all Java schedule events

Classification	Java supporting function		
Description	Delete all Java schedule events	Function name	Elib_SCH_Java_All_Reset
Functional overview	<p>- This function deletes all Java schedules registered by using "Register Java schedule event."</p>		
Include file	srv_sch.h		
Prototype	Int Elib_SCH_Java_All_Reset(Ap_ID);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Return value	Type	I/O	Description
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure: Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.43 Read Java schedule event

Classification	Java supporting function			
Description	Read event	Java schedule	Function name	Elib_SCH_Java_Read
Functional overview	<ul style="list-style-type: none"> - This function reads a Java schedule with a specified schedule number. - This function sets the schedule number into the Java schedule data structure represented by the schedule data readout area. The result is put into the said structure. - Specifying an unregistered schedule number returns "Unregistered." - As the result of a readout, the date and time registered for it is returned regardless of the repetition setting. 			
Include file	srv_sch.h			
Prototype	Int Elib_SCH_Java_Read(Ap_ID, Sch_Data);			
Argument	Type	I/O	Description	
Ap_ID	unsigned int	I	Application ID	
Sch_Data	_ELIB_SCH_JAVA_DATA *	I/O	Schedule data readout area	
Return value	Type	I/O	Description	
Ret	Int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered data : ELIB_SCH_NODATA	
Remark				
<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>				

5.44 Register Java schedule event

Classification	Java supporting function		
Description	Register Java schedule event (API)	Function name	Elib_SCH_Java_Set_API
Functional overview	<ul style="list-style-type: none"> - Applications call this interface (function) to register a Java schedule (API). - If the number of already registered Java schedule events has reached the maximum, "Number of events registered reaching maximum" is returned then this function quits. - Set data into the Java schedule data structure, then call this interface (function). (For details about the structure, see "4. List of Structures.") - For Index (Index_No), specify a value between 1 and 4. Otherwise, the "parameter error" is returned and the function quits. - If there is a Java schedule already registered with the "entry ID" and "index" specified in arguments, the former one is overwritten. Otherwise, the Java schedule event is newly registered. - The specifiable schedule types are as follows: <ul style="list-style-type: none"> ELIB_SCH_SINGLE /* One-off schedule */ ELIB_SCH_DAY /* Daily schedule */ ELIB_SCH_WEEKSLCT /* Weekly schedule */ - To set "weekly schedule" to the schedule type, take a logical sum (OR) of all specified days of the week then set it into the "day of the week selection (WeekSlct)" member. - For the values of the respective days of the week, use the following definitions. In addition, to select all the days of the week, set "ELIB_SCH_B_WEEKALL": <ul style="list-style-type: none"> ELIB_SCH_B_SUN /* Sunday */ ELIB_SCH_B_MON /* Monday */ ELIB_SCH_B_TUE /* Tuesday */ ELIB_SCH_B_WED /* Wednesday */ ELIB_SCH_B_THU /* Thursday */ ELIB_SCH_B_FRI /* Friday */ ELIB_SCH_B_SAT /* Saturday */ 		

ELIB_SCH_B_WEEKALL /* All days from Sunday to Saturday */

- When the schedule type is "weekly schedule" and if no day of the week is specified for the "day of the week selection (WeekSlt)" member, a "parameter error" is returned.
- When the schedule type is "weekly schedule," the date on which the alarm is emitted next time is calculated. If the result of calculation is beyond the end date and time the calendar can support, "weekly specified setting out of scope (ELIB_SCH_WEEKOVERFLOW)" is returned and this function quits.
- When the schedule type is "weekly schedule," the date is calculated as stated in (1) or (2) then output to the "year, month, and day" members of the 3rd argument (Java schedule data structure):
 - (1) When the date input to the data is before the system date and "day of the week selection" is applied to the day of the week of the system date. → The system date is output.
 - (2) Other than (1) → The date on which the alarm is emitted next time is calculated and the result of calculation is output.

(Note) The "year, month, and day" are to be output to the 2nd argument, the day of the week of the date is output to the "day of the week" member.
- When "one-off" is set to the schedule type, the alarm for the one-off schedule is not to be emitted again by any means once it is emitted even if the time to emit the alarm comes again after the system time is changed.
- When the specified schedule type is "daily" or "weekly," the date on which the alarm is to be emitted is not changed even if the system time is changed to an earlier timer.
- The following shows an example of a case where the system time is changed to an earlier time:

[Registered Java schedule data (API)]

Date and time : December 1, (Monday) 16:00

Repetition type : Repeated daily

[System time]

December 4, 2003 (Thursday) 15:00

[Next alarming date and time]

December 4, 2003 (Thursday) 16:00

↓

When the system time is changed to an earlier time while there are data indicated above:

↓

[Registered Java schedule data (API)]

Date and time : December 1, 2003 (Monday) 16:00

Repetition type : Repeated daily

[System time]

December 2, 2002 (Tuesday) 15:00

[Next alarming date and time]

December 4, 2003 (Thursday) 16:00 (*)

(*) When the system time is changed to an earlier time, the next alarming date and time is not calculated.

- When the schedule type is set to "daily" or "weekly" and if the system time is changed to a later time, the alarming date and time is to be changed:

- The following shows an example of a case where the system time is changed to a later time:

[Registered Java schedule data (API)]

Date and time : December 1, 2003 (Monday) 16:00

Repetition type : Repeated daily

[System time]

December 4, 2003 (Thursday) 15:00

[Next alarming date and time]

December 4, 2003 (Thursday) 16:00

↓

When the system time is changed to a later time while there are data indicated above:

↓

[Registered Java schedule data (API)]

Date and time : December 1, 2003 (Monday) 16:00

Repetition type : Repeated daily

[System time]

December 6, 2002 (Saturday) 15:00

[Next alarming date and time]

December 6, 2003 (Saturday) 16:00 (*)

(*) When the system time is changed to a later time, the next alarming date and time is calculated.

- For the date and time, set the "year, month, day, hour, minute, and day of the week."
- 0 is always set to "second."
- For the alarm type, set "alarming enabled."

- To receive notifications about an event specified by this interface (function), the application needs to make a call to the "A-19 Start posting alarm notifications" function when it starts up to inform that the application is ready to hand the event. (For the format of events posted from Schedule Service to service-requesting applications, see "A-1 Start requesting Schedule Service event notifications.")

Include file	srv_sch.h		
Prototype	Int Elib_SCH_Java_Set_API(Ap_ID, Entry_ID, Index_No, Sch_Data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Entry_ID	long	I	Entry ID (i-appli ID)
Index_No	unsigned char	I	Index
Sch_Data	_ELIB_SCH_JAVA_DATA *	I/O	Schedule data to be registered
Return value	Type	I/O	Description
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered data : ELIB_SCH_NODATA Max registered Events reached : ELIB_SCH_MAXDATA Weekly specified setting out of scope : ELIB_SCH_WEEKOVERFLOW Not supported year : ELIB_SCH_OVERFLOW
Remark			
<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>			

5.45 Delete Java schedule event

Classification	Java supporting function		
Description	Delete Java schedule event (API)	Function name	Elib_SCH_Java_Reset_API
Functional overview	<p>- This function deletes a Java schedule registered by using "Register Java schedule event (API)."</p> <p>- In arguments, specify the "entry ID" and "index" of a Java schedule to be deleted.</p> <p>- For the 3rd argument "Index," specify a value between 1 and 4. Otherwise, the "parameter error" is returned and the function quits.</p> <p>- If there is no data with the "entry ID" and "index" specified in arguments, "Unregistered" is returned and this function quits.</p>		
Include file	srv_sch.h		
Prototype	Int Elib_SCH_Java_Reset_API(Ap_ID, Entry_ID, Index_No);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Entry_ID	Long	I	Entry ID (i-appli ID)
Index_No	unsigned char	I	Index
Return value	Type	I/O	Description
ret	Int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered data : ELIB_SCH_NODATA
Remark			
<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>			

5.46 Delete all Java schedule events

Classification	Java supporting function		
Description	Delete all Java schedule events (API)	Function name	Elib_SCH_Java_All_Reset_API
Functional overview			
<p>- This function deletes all Java schedules registered by using "E-5 Register Java schedule event (API)."</p>			
Include file	srv_sch.h		
Prototype	Int Elib_SCH_Java_All_Reset_API(Ap_ID);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Return value	Type	I/O	Description
Ret	Int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered data : ELIB_SCH_NODATA
Remark			
<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>			

5.47 Read Java schedule event

Classification	Java supporting function		
Description	Read Java schedule event (API)	Function name	Elib_SCH_Java_Read_API
Functional overview	<ul style="list-style-type: none"> - This function reads a Java schedule registered by using "Register Java schedule event (API)." - Specify the "entry ID" and "index" of a Java schedule to be read in the arguments. - For the "Index" argument, specify a value between 1 and 4. Otherwise, the "parameter error" is returned and the function quits. - If no data registered with the "entry ID" and "index" same as those specified in the arguments, this function returns "Unregistered" and quits. - As the result of a readout, the date and time registered for it is returned regardless of the repetition setting. 		
Include file	srv_sch.h		
Prototype	int Elib_SCH_Java_Read_API(Ap_ID, Sch_Data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Sch_Data	_ELIB_SCH_JAVA_DATA *	I/O	Schedule data readout buffer pointer
Return value	Type	I/O	Description
ret	Int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered data : ELIB_SCH_NODATA
Remark			
Necessary procedure: Create a MSB object by using MsbObjectCreate in respective applications.			

5.48 Read inactivated Java schedules from list

Classification	Java supporting function		
Description	Read Java schedules inactivated to list	Function name	Elib_SCH_Java_NotRun_List_Read
Functional overview	<ul style="list-style-type: none"> - This function sorts Java schedules whose time to be automatically activated passed while the power is off in order of registrations then reads them one by one. - The application calls this interface (function) once for each Java schedules to be read until ELIB_SCH_NODATA is returned in the return value. - A Java schedule to be read includes repetitions of it. - This interface (function) reads data of Java schedules, listing and sorting them when the listing number is set to the initial value 0. - The read data are put into the Java schedule data structure. (The data are read with the time it was inactivated.) - In the second or later call to this interface (function), specify the listing number of a Java schedule event to be read. - The list of inactivated Java schedules created by this function remains maintained after the power being turned off unless the "Clear list of Java schedules inactivated" function is called. Therefore, to check the inactivated Java schedules, call "E-10 Clear list of Java schedules inactivated" function. 		
Include file	srv_sch.h		
Protoytype	Int Elib_SCH_Java_NotRun_List_Read(Ap_ID, No, Sch_Data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
No	int	I	Listing number
Sch_Data	_ELIB_SCH_JAV A_DATA *	O	Schedule data readout buffer pointer
Return value	Type	I/O	Description
Ret	int	O	Successful : Number of events registered in the list Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered data : ELIB_SCH_NODATA
Remark			

Necessary procedure:

Create a MSB object by using MsbObjectCreate in respective applications.

5.49 Clear list of inactivated Java schedules

Classification	Java supporting function		
Description	Clear list of inactivated Java schedules	Prototype	Elib_SCH_Java_NotRun_List_StatusChange
Functional overview	<ul style="list-style-type: none"> - This function clears the list of Java schedules whose time to be automatically started up passed while the power is off. - The application can execute this function whenever an inactivated icon is displayed. - If this function is not called (while an inactivated icon is displayed), Java schedule data registered in the list of inactivated Java schedules remain in the list when the power is turned on next time. 		
Include file	srv_sch.h		
Prototype	Int Elib_SCH_Java_NotRun_List_StatusChange(Ap_ID);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Return value	Type	I/O	Description
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.50 Register holiday

Classification	Holiday setting support function																																				
Description	Register holiday	Function name	Elib_SCH_Holiday_Set																																		
Functional overview	<ul style="list-style-type: none"> - The number of holiday data allowed for the user to register must be 100. - Only one holiday can be registered per day. However, it is allowed to register a holiday to a day with a default holiday already set. (For the default holidays, see "Sheet Attached 1 List of Default Holidays.") - This function sets the data to be registered into the holiday data structure represented by the holiday data to be registered. The result is put into the said structure. (For details about the structure, see "4. List of Structures.") - Holiday data to be registered (holiday data structure) <table> <tr> <th>Data item name</th><th>Value</th><th>IN/OUT</th><th>Remark</th></tr> <tr> <td>Application ID</td><td>Application ID</td><td>IN</td><td></td></tr> <tr> <td>Year</td><td>Christian era</td><td>IN</td><td>Christian year in the supported calendars</td></tr> <tr> <td>Month</td><td>1 to 12</td><td>IN</td><td></td></tr> <tr> <td>Day</td><td>1 to 31</td><td>IN</td><td></td></tr> <tr> <td rowspan="7">Day of the week</td><td>ELIB_SCH_SUN (Sunday)</td><td rowspan="7">OUT</td><td rowspan="7"></td></tr> <tr> <td>ELIB_SCH_MON (Monday)</td></tr> <tr> <td>ELIB_SCH_TUE (Tuesday)</td></tr> <tr> <td>ELIB_SCH_WED (Wednesday)</td></tr> <tr> <td>ELIB_SCH_THU (Thursday)</td></tr> <tr> <td>ELIB_SCH_FRI (Friday)</td></tr> <tr> <td>ELIB_SCH_SAT (Saturday)</td></tr> <tr> <td>Holiday number</td><td>0: Register new</td><td>IN/OUT</td><td>Any setting out of the scope</td></tr> </table>			Data item name	Value	IN/OUT	Remark	Application ID	Application ID	IN		Year	Christian era	IN	Christian year in the supported calendars	Month	1 to 12	IN		Day	1 to 31	IN		Day of the week	ELIB_SCH_SUN (Sunday)	OUT		ELIB_SCH_MON (Monday)	ELIB_SCH_TUE (Tuesday)	ELIB_SCH_WED (Wednesday)	ELIB_SCH_THU (Thursday)	ELIB_SCH_FRI (Friday)	ELIB_SCH_SAT (Saturday)	Holiday number	0: Register new	IN/OUT	Any setting out of the scope
Data item name	Value	IN/OUT	Remark																																		
Application ID	Application ID	IN																																			
Year	Christian era	IN	Christian year in the supported calendars																																		
Month	1 to 12	IN																																			
Day	1 to 31	IN																																			
Day of the week	ELIB_SCH_SUN (Sunday)	OUT																																			
	ELIB_SCH_MON (Monday)																																				
	ELIB_SCH_TUE (Tuesday)																																				
	ELIB_SCH_WED (Wednesday)																																				
	ELIB_SCH_THU (Thursday)																																				
	ELIB_SCH_FRI (Friday)																																				
	ELIB_SCH_SAT (Saturday)																																				
Holiday number	0: Register new	IN/OUT	Any setting out of the scope																																		

			specified holiday number is assigned.
Repetition setting	ELIB_SCH_SINGLE (one-off)	IN	
	ELIB_SCH_YEAR (annual)		
Holiday content	Text of holiday content	IN	

- The day of the week is calculated by this function according to the date and set to the day of the week member in the structure delivered as an argument.
- The size of the holiday content is 20 bytes.
(Schedule Service handles the holiday content with the fixed size of 20 bytes, thus the size of the holiday content size needs to be controlled by the user.)
- For a new holiday registered successfully, the "holiday number" is automatically set.
- After the "holiday number" is automatically set, set it to the holiday number member in the structure delivered as an argument.
- The "year, month, and day" specified for the date and time are to be checked for the scope. For the scope, see "6. Calendar Coverage."
- If there are holiday data already registered to the date, the existing data of the holiday are deleted.
See "Check another holiday on identical date.")

Include file	srv_sch.h		
Prototype	Int Elib_SCH_Holiday_Set(Ap_ID, hldaydata_p);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
hldaydata_p	ELIB_SCH_HOLIDAY_DATA *	I/O	Holiday data to be registered
Return value	Type	I/O	Description
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Max registered Events reached : ELIB_SCH_MAXDATA
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.51 Delete holiday

Classification	Holiday setting support function																																									
Description	Delete holiday	Function name	Elib_SCH_Holiday_Delete																																							
Functional overview	<ul style="list-style-type: none"> - This function deletes a holiday with a specified number or date. - This function sets the number or date of the holiday to be deleted into the holiday data structure represented by the holiday data to be deleted. - Holiday data to be deleted (holiday data structure) <table> <tr> <th>Data item name</th><th>Value</th><th>IN/OUT</th><th>Remark</th></tr> <tr> <td>Application ID</td><td>Application ID</td><td>IN</td><td></td></tr> <tr> <td>Year</td><td>Christian era</td><td>IN</td><td>Christian year in the supported calendars</td></tr> <tr> <td>Month</td><td>1 to 12</td><td>IN</td><td></td></tr> <tr> <td>Day</td><td>1 to 31</td><td>IN</td><td></td></tr> <tr> <td rowspan="7">Day of the week</td><td>ELIB_SCH_SUN (Sunday)</td><td rowspan="7">IN</td><td rowspan="7">Unused</td></tr> <tr> <td>ELIB_SCH_MON (Monday)</td></tr> <tr> <td>ELIB_SCH_TUE (Tuesday)</td></tr> <tr> <td>ELIB_SCH_WED (Wednesday)</td></tr> <tr> <td>ELIB_SCH_THU (Thursday)</td></tr> <tr> <td>ELIB_SCH_FRI (Friday)</td></tr> <tr> <td>ELIB_SCH_SAT (Saturday)</td></tr> <tr> <td rowspan="2">Holiday number</td><td>0: Date specified</td><td rowspan="2">IN</td><td rowspan="2">Any setting out of the scope is a parameter error. If the Date is passed as parameter, data registered for the specified date are deleted. If the registered number is passed as parameter, data registered for the specified holiday number are deleted.</td></tr> <tr> <td>1 to 115 : Registered number specified (101 to 115 are the holiday numbers of the default holidays.)</td></tr> <tr> <td>Repetition setting</td><td>ELIB_SCH_SINGLE (one-off)</td><td>IN</td><td>Unused</td></tr> </table>			Data item name	Value	IN/OUT	Remark	Application ID	Application ID	IN		Year	Christian era	IN	Christian year in the supported calendars	Month	1 to 12	IN		Day	1 to 31	IN		Day of the week	ELIB_SCH_SUN (Sunday)	IN	Unused	ELIB_SCH_MON (Monday)	ELIB_SCH_TUE (Tuesday)	ELIB_SCH_WED (Wednesday)	ELIB_SCH_THU (Thursday)	ELIB_SCH_FRI (Friday)	ELIB_SCH_SAT (Saturday)	Holiday number	0: Date specified	IN	Any setting out of the scope is a parameter error. If the Date is passed as parameter, data registered for the specified date are deleted. If the registered number is passed as parameter, data registered for the specified holiday number are deleted.	1 to 115 : Registered number specified (101 to 115 are the holiday numbers of the default holidays.)	Repetition setting	ELIB_SCH_SINGLE (one-off)	IN	Unused
Data item name	Value	IN/OUT	Remark																																							
Application ID	Application ID	IN																																								
Year	Christian era	IN	Christian year in the supported calendars																																							
Month	1 to 12	IN																																								
Day	1 to 31	IN																																								
Day of the week	ELIB_SCH_SUN (Sunday)	IN	Unused																																							
	ELIB_SCH_MON (Monday)																																									
	ELIB_SCH_TUE (Tuesday)																																									
	ELIB_SCH_WED (Wednesday)																																									
	ELIB_SCH_THU (Thursday)																																									
	ELIB_SCH_FRI (Friday)																																									
	ELIB_SCH_SAT (Saturday)																																									
Holiday number	0: Date specified	IN	Any setting out of the scope is a parameter error. If the Date is passed as parameter, data registered for the specified date are deleted. If the registered number is passed as parameter, data registered for the specified holiday number are deleted.																																							
	1 to 115 : Registered number specified (101 to 115 are the holiday numbers of the default holidays.)																																									
Repetition setting	ELIB_SCH_SINGLE (one-off)	IN	Unused																																							

	ELIB_SCH_YEAR (annual)		
Holiday content	Text of holiday content	IN	Unused

- If the holiday data to be deleted have not been registered, this function returns with "Unregistered."
- Setting 0 to the holiday number makes reference to the year, month, and day in the holiday data structure and delete data registered with the date. (However, the default holidays are not subject to date-specified deletion.)
- When a date is specified together with "registered number specified," the specified date is ignored.
- For the date and time, set only "year, month, and day."

Include file	Srv_sch.h		
Prototype	Int Elib_SCH_Holiday_Delete(Ap_ID, holiday_data_p);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
holiday_data_p	_ELIB_SCH_HOLIDAY_DATA *	I	Holiday data to be deleted
Return value	Type	I/O	Description
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered data : ELIB_SCH_NODATA
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.52 Delete all holidays

Classification	Holiday setting support function		
Description	Delete all holidays	Function name	Elib_SCH_Holiday_All_Delete
Functional overview	<p>- This function deletes all user-defined holidays.</p>		
Include file	srv_sch.h		
Prototype	Int Elib_SCH_Holiday_All_Delete(Ap_ID);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Return value	Type	I/O	Description
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure: Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.53 Delete all holidays before specified date

Classification	Holiday setting support function		
Description	Delete all holidays before specified date	Function name	Elib_SCH_Holiday_Past_Delete
Functional overview	<div><div><div><div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div></</div></div></div></div></div>		

5.54 Read holiday

Classification	Holiday setting support function																																					
Description	Read holiday	Function name	Elib_SCH_Holiday_Read																																			
Functional overview	<p>- This function sets the holiday number into the holiday data structure represented by the user-defined holiday data readout area. The result is set to the said structure and default holiday data readout area.</p> <p>- User-defined/default holiday data (holiday data structure)</p> <table> <tr> <th>Data item name</th><th>Value</th><th>IN/OUT</th><th>Remark</th></tr> <tr> <td>Application ID</td><td>Application ID</td><td>OUT</td><td></td></tr> <tr> <td>Year</td><td>Christian era</td><td>IN/OUT</td><td>Christian year in the supported calendars</td></tr> <tr> <td>Month</td><td>1 to 12</td><td>IN/OUT</td><td></td></tr> <tr> <td>Day</td><td>1 to 31</td><td>IN/OUT</td><td></td></tr> <tr> <td rowspan="7">Day of the week</td><td>ELIB_SCH_SUN (Sunday)</td><td rowspan="7">OUT</td><td rowspan="7"></td></tr> <tr> <td>ELIB_SCH_MON (Monday)</td></tr> <tr> <td>ELIB_SCH_TUE (Tuesday)</td></tr> <tr> <td>ELIB_SCH_WED (Wednesday)</td></tr> <tr> <td>ELIB_SCH_THU (Thursday)</td></tr> <tr> <td>ELIB_SCH_FRI (Friday)</td></tr> <tr> <td>ELIB_SCH_SAT (Saturday)</td></tr> <tr> <td rowspan="2">Holiday number</td><td>0: Date specified</td><td rowspan="2">IN/OUT</td><td rowspan="2">Any setting out of the scope is a parameter error. If the Date is passed as parameter, data registered for the specified date are deleted. If the registered number is passed as parameter, data registered for the specified</td></tr> <tr> <td>1 to 115 : Registered number specified (101 to 115 are the holiday numbers of the default holidays.)</td></tr> </table>			Data item name	Value	IN/OUT	Remark	Application ID	Application ID	OUT		Year	Christian era	IN/OUT	Christian year in the supported calendars	Month	1 to 12	IN/OUT		Day	1 to 31	IN/OUT		Day of the week	ELIB_SCH_SUN (Sunday)	OUT		ELIB_SCH_MON (Monday)	ELIB_SCH_TUE (Tuesday)	ELIB_SCH_WED (Wednesday)	ELIB_SCH_THU (Thursday)	ELIB_SCH_FRI (Friday)	ELIB_SCH_SAT (Saturday)	Holiday number	0: Date specified	IN/OUT	Any setting out of the scope is a parameter error. If the Date is passed as parameter, data registered for the specified date are deleted. If the registered number is passed as parameter, data registered for the specified	1 to 115 : Registered number specified (101 to 115 are the holiday numbers of the default holidays.)
Data item name	Value	IN/OUT	Remark																																			
Application ID	Application ID	OUT																																				
Year	Christian era	IN/OUT	Christian year in the supported calendars																																			
Month	1 to 12	IN/OUT																																				
Day	1 to 31	IN/OUT																																				
Day of the week	ELIB_SCH_SUN (Sunday)	OUT																																				
	ELIB_SCH_MON (Monday)																																					
	ELIB_SCH_TUE (Tuesday)																																					
	ELIB_SCH_WED (Wednesday)																																					
	ELIB_SCH_THU (Thursday)																																					
	ELIB_SCH_FRI (Friday)																																					
	ELIB_SCH_SAT (Saturday)																																					
Holiday number	0: Date specified	IN/OUT	Any setting out of the scope is a parameter error. If the Date is passed as parameter, data registered for the specified date are deleted. If the registered number is passed as parameter, data registered for the specified																																			
	1 to 115 : Registered number specified (101 to 115 are the holiday numbers of the default holidays.)																																					

Repetition setting	ELIB_SCH_SINGLE (one-off)	OUT	
	ELIB_SCH_YEAR (annual)		
Holiday content	Text of holiday content	OUT	

- If either holiday data or default holiday data are to be acquired, specify both data buffer (2nd and 3rd arguments).
- If the holiday data to be read have not been registered, this function returns with "Unregistered."
- Specifying 0 to the holiday number makes reference to the year, month, and day in the holiday data structure and reads the data registered to the date.
- In a date-specified read, if there are both user-defined holiday and default holiday on the date, both the holidays are read.
- When a date is specified together with "registered number specified," the specified date is ignored.
- If either user-defined holiday or default holiday can be successfully read, the normal end shall be returned.
- If neither user-defined holiday nor default holiday can be read, the storage area is cleared with zeros.
- The day of the week of data to be read is calculated by this function according to its date and set to the day of the week member in the structure delivered as an argument.
- For the date and time, set only "year, month, and day."

Include file	srv_sch.h		
Prototype	Int Elib_SCH_Holiday_Read(Ap_ID, hldaydata_p, dflt_hldaydata_p);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
hldaydata_p	ELIB_SCH_HOLIDAY_DATA *	I/O	User-defined holiday data readout area
dflt_hldaydata_p	ELIB_SCH_HOLIDAY_DATA *	O	Default holiday data readout area
Return value	Type	I/O	Description
Ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered data : ELIB_SCH_NODATA
Remark			

Necessary procedure:

Create a MSB object by using MsbObjectCreate in respective applications.

5.55 Read holiday list

Classification	Holiday setting support function																																													
Description	Read holiday list	Function name	Elib_SCH_Holiday_ListData_Read																																											
Functional overview	<ul style="list-style-type: none"> - This function sorts registered holidays in chronological order then reads them one by one. - The application calls this interface (function) once for each of holidays to be read. The readout data are set into the holiday data structure represented by the read holiday data area. - Read holiday data area (holiday data structure) <table> <tr> <th>Data item name</th><th>Value</th><th>IN/OUT</th><th>Remark</th></tr> <tr> <td>Application ID</td><td>Application ID</td><td>OUT</td><td></td></tr> <tr> <td>Year</td><td>Christian era</td><td>OUT</td><td>Christian year in the supported calendars</td></tr> <tr> <td>Month</td><td>1 to 12</td><td>OUT</td><td></td></tr> <tr> <td>Day</td><td>1 to 31</td><td>OUT</td><td></td></tr> <tr> <td rowspan="7">Day of the week</td><td>ELIB_SCH_SUN (Sunday)</td><td rowspan="7">OUT</td><td rowspan="7"></td></tr> <tr> <td>ELIB_SCH_MON (Monday)</td></tr> <tr> <td>ELIB_SCH_TUE (Tuesday)</td></tr> <tr> <td>ELIB_SCH_WED (Wednesday)</td></tr> <tr> <td>ELIB_SCH_THU (Thursday)</td></tr> <tr> <td>ELIB_SCH_FRI (Friday)</td></tr> <tr> <td>ELIB_SCH_SAT (Saturday)</td></tr> <tr> <td>Holiday number</td><td>1 to 100</td><td>OUT</td><td></td></tr> <tr> <td rowspan="2">Repetition setting</td><td>ELIB_SCH_SINGLE (one-off)</td><td rowspan="2">OUT</td><td rowspan="2"></td></tr> <tr> <td>ELIB_SCH_YEAR (annual)</td></tr> <tr> <td>Holiday content</td><td>Text of holiday content</td><td>OUT</td><td></td></tr> </table> <ul style="list-style-type: none"> - The day of the week of data to be read is calculated by this function according to its date and set to the day of the week member in the structure delivered as an argument. 			Data item name	Value	IN/OUT	Remark	Application ID	Application ID	OUT		Year	Christian era	OUT	Christian year in the supported calendars	Month	1 to 12	OUT		Day	1 to 31	OUT		Day of the week	ELIB_SCH_SUN (Sunday)	OUT		ELIB_SCH_MON (Monday)	ELIB_SCH_TUE (Tuesday)	ELIB_SCH_WED (Wednesday)	ELIB_SCH_THU (Thursday)	ELIB_SCH_FRI (Friday)	ELIB_SCH_SAT (Saturday)	Holiday number	1 to 100	OUT		Repetition setting	ELIB_SCH_SINGLE (one-off)	OUT		ELIB_SCH_YEAR (annual)	Holiday content	Text of holiday content	OUT	
Data item name	Value	IN/OUT	Remark																																											
Application ID	Application ID	OUT																																												
Year	Christian era	OUT	Christian year in the supported calendars																																											
Month	1 to 12	OUT																																												
Day	1 to 31	OUT																																												
Day of the week	ELIB_SCH_SUN (Sunday)	OUT																																												
	ELIB_SCH_MON (Monday)																																													
	ELIB_SCH_TUE (Tuesday)																																													
	ELIB_SCH_WED (Wednesday)																																													
	ELIB_SCH_THU (Thursday)																																													
	ELIB_SCH_FRI (Friday)																																													
	ELIB_SCH_SAT (Saturday)																																													
Holiday number	1 to 100	OUT																																												
Repetition setting	ELIB_SCH_SINGLE (one-off)	OUT																																												
	ELIB_SCH_YEAR (annual)																																													
Holiday content	Text of holiday content	OUT																																												

- A holiday to be read includes repetitions of it. However, no default holidays are included.
- This function lists holidays at the initial call (made with 0 set to the listing number argument). In the second or later call, specifying the position of a holiday in the listing number argument allows the holiday data to be read. Note that the initial call returns the holiday at the top of the list.
- When the repetition setting is set to "annual," the nearest future day (date on which the event is repeated next) is set to the holiday data structure and output. However, if the nearest future day is beyond the end date and time the calendar can support, an allowable latest day is set to the holiday data structure and output. For the end date and time the calendar can support, see "6. Calendar Coverage."

Include file	srv_sch.h		
Prototype	Int Elib_SCH_Holiday_ListData_Read (Ap_ID, holiday_no, hldaydata_p);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
holiday_no	int	I	Listing number
hldaydata_p	_ELIB_SCH_HOLIDAY_DATA *	O	Read holiday data area
Return value	Type	I/O	Description
ret	int	O	Successful : Number of events registered in the list Unsuccessful : ELIB_SCH_NG Incorrect parameter: ELIB_SCH_PARAM_NG Unregistered data : ELIB_SCH_NODATA
Remark			

Necessary procedure:

Create a MSB object by using MsbObjectCreate in respective applications.

Note:

The information put in the list by the initial call comes from static areas of functions. Therefore, the caller must note

that an initial call made before it reads all holiday data modifies static areas.

5.56 Read number of registered holidays

Classification	Holiday setting support function		
Description	Read number of registered holidays	Function name	Elib_SCH_Holiday_Total_Num_Read
Functional overview	<ul style="list-style-type: none"> - This function gets the number of already registered holidays. - The number of only user-registered holidays is acquired exclusive of the default holidays. 		
Include file	srv_sch.h		
Prototype	Int Elib_SCH_Holiday_Total_Num_Read(Ap_ID);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Return value	Type	I/O	Description
ret	int	O	Successful : Number of registered holidays (0 to 100) Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	Necessary procedure: Create an MSB object by using MsbObjectCreate in respective applications.		

5.57 Check another holiday on identical date

Classification	Holiday setting support function																																
Description	Check another holiday on identical date	Function name	Elib_SCH_Holiday_Same_Date_Check																														
Functional overview	<div><ul style="list-style-type: none">- This function checks whether there is another holiday already registered to the date of a specified holiday.- This function makes a check by using the "date specified, repetition setting, and registration number."- This function makes a check on combinations of repetition types as follows:</div> <table><tr><th>Holiday 1</th><th>Holiday 2</th><th>Condition 1</th><th>Condition 2</th><th>Result</th></tr><tr><td rowspan="6">One-off</td><td rowspan="2">One-off</td><td>Year, Month, Day ≠ Year, Month, Day</td><td>—</td><td>No other on the date</td></tr><tr><td>Year, Month, Day = Year, Month, Day</td><td>—</td><td>Another on the date</td></tr><tr><td rowspan="3">Annual</td><td>Month and Day of one-off ≠ Month and Day of annual</td><td>—</td><td>No other on the date</td></tr><tr><td rowspan="2">Month and Day of one-off = Month and Day of annual</td><td>Date (year) of one-off < Date (year) of annual</td><td>No other on the date</td></tr><tr><td>Date (year) of annual ≤ Date (year) of one-off</td><td>Another on the date</td></tr><tr><td rowspan="2">Annual</td><td rowspan="2">Annual</td><td>Month and Day ≠ Month and Day</td><td>—</td><td>No other on the date</td></tr><tr><td>Month and Day = Month and Day</td><td>—</td><td>Another on the date</td></tr></table> <div><ul style="list-style-type: none">- The default holidays are not subject to be checked by this function. (A user-defined holiday can be registered to a date with a default holiday set.)</div>			Holiday 1	Holiday 2	Condition 1	Condition 2	Result	One-off	One-off	Year, Month, Day ≠ Year, Month, Day	—	No other on the date	Year, Month, Day = Year, Month, Day	—	Another on the date	Annual	Month and Day of one-off ≠ Month and Day of annual	—	No other on the date	Month and Day of one-off = Month and Day of annual	Date (year) of one-off < Date (year) of annual	No other on the date	Date (year) of annual ≤ Date (year) of one-off	Another on the date	Annual	Annual	Month and Day ≠ Month and Day	—	No other on the date	Month and Day = Month and Day	—	Another on the date
Holiday 1	Holiday 2	Condition 1	Condition 2	Result																													
One-off	One-off	Year, Month, Day ≠ Year, Month, Day	—	No other on the date																													
		Year, Month, Day = Year, Month, Day	—	Another on the date																													
	Annual	Month and Day of one-off ≠ Month and Day of annual	—	No other on the date																													
		Month and Day of one-off = Month and Day of annual	Date (year) of one-off < Date (year) of annual	No other on the date																													
			Date (year) of annual ≤ Date (year) of one-off	Another on the date																													
	Annual	Annual	Month and Day ≠ Month and Day	—	No other on the date																												
Month and Day = Month and Day			—	Another on the date																													
Include file	srv_sch.h																																
Prototype	Int Elib_SCH_Holiday_Same_Date_Check(Ap_ID, hldaydata_p);																																
Argument	Type	I/O	Description																														
Ap_ID	unsigned int	I	Application ID																														
hldaydata_p	_ELIB_SCH_HOLIDAY_DATA *	I	Holiday to be checked																														
Return value	Type	I/O	Description																														
ret	int	O	Registerable : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Another on the date : ELIB_SCH_SAMEDATE																														

			Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

			ELIB_SCH_NODATA
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.59 Reset default holidays

Classification	Holiday setting support function		
Description	Reset default holidays	Function name	Elib_SCH_Default_Holiday_Reset
Functional overview	<p>- This function registers (initializes) the default 15 holidays.</p> <p>* For the default holidays, see "Sheet Attached 1 List of Default Holidays."</p>		
Include file	srv_sch.h		
Prototype	Int Elib_SCH_Default_Holiday_Reset (Ap_ID);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Return value	Type	I/O	Description
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.60 Get default holidays (in specified year/month)

Classification	Holiday setting support function		
Description	Get default holidays (in specified year/month)	Function name	Elib_SCH_Default_Holiday_Read
Functional overview			

- This function checks whether there are any default holidays in a specified target year and month.

Target year and month (date of holiday data structure)

Data item name	Value	Remark
Year	Christian era	Christian year of the calendar coverage
Month	1 to 12	
Day		Unused
Day of the week		Unused

- The setting status is put into the holiday information storage array.
(The holiday information storage array consists of 31 elements. Elements 0 to 30 correspond to the 1st day to 31st day of a month.)

<Example>

```
unsigned char  info[ELIB_SCH_HAM_DAYPLAN_INFO] ;
```

info[0]

- Stores the setting status about the 1st day of the specified year and month.

info[1]

- Stores the setting status about the 2nd day of the specified year and month.

:

info[27]

- Stores the setting status about the 28th day of the specified year and month.

info[28]

- Stores the setting status about the 29th day of the specified year and month.
(stores 0 if there is no 29th day in the specified year and month.)

info[29]

- Stores the setting status about the 30th day of the specified year and month.
(stores 0 if there is no 30th day in the specified year and month.)

info[30]

- Stores the setting status about the 31st day of the specified year and month.
(stores 0 if there is no 31st day in the specified year and month.)

- For a default holiday, the following value is set:

ELIB_SCH_DEFAULT_HOLIDAY_EXIST

/* Default holiday

*/

- The value of the target year and month is to be checked for the scope. For the scope, see "6. Calendar Coverage."

Include file	srv_sch.h		
Prototype	Int Elib_SCH_Default_Holiday_Read(Ap_ID, date_p, info);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
date_p	_ELIB_SCH_HOLIDAY_CAL_DATA *	I	Target year and month
Info	unsigned char *	O	Holiday information storage array
Return value	Type	I/O	Description
Ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered data : ELIB_SCH_NODATA
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.61 Register anniversary

Classification	Anniversary supporting function																																				
Description	Register anniversary	Function name	Elib_SCH_Anniversary_Set																																		
Functional overview	<ul style="list-style-type: none"> - The number of anniversary data allowed for the user to register must be 100. - Only one anniversary can be registered per day. - This function sets the data to be registered into the anniversary data to be registered represented by the anniversary data to be registered. The result is put into the said structure. (For details about the structure, see "4. List of Structures.") <p>Anniversary data to be registered (anniversary data structure)</p> <table> <tr> <th>Data item name</th><th>Value</th><th>IN/O UT</th><th>Remark</th></tr> <tr> <td>Application ID</td><td>Application ID</td><td>IN</td><td></td></tr> <tr> <td>Year</td><td>Christian era</td><td>IN</td><td>Christian year in the supported calendars</td></tr> <tr> <td>Month</td><td>1 to 12</td><td>IN</td><td></td></tr> <tr> <td>Day</td><td>1 to 31</td><td>IN</td><td></td></tr> <tr> <td rowspan="7">Day of the week</td><td>ELIB_SCH_SUN (Sunday)</td><td rowspan="7">OUT</td><td rowspan="7"></td></tr> <tr> <td>ELIB_SCH_MON (Monday)</td></tr> <tr> <td>ELIB_SCH_TUE (Tuesday)</td></tr> <tr> <td>ELIB_SCH_WED (Wednesday)</td></tr> <tr> <td>ELIB_SCH_THU (Thursday)</td></tr> <tr> <td>ELIB_SCH_FRI (Friday)</td></tr> <tr> <td>ELIB_SCH_SAT (Saturday)</td></tr> <tr> <td>Anniversary</td><td>0: Register new</td><td>IN/O</td><td>Any setting out of the scope is a</td></tr> </table>			Data item name	Value	IN/O UT	Remark	Application ID	Application ID	IN		Year	Christian era	IN	Christian year in the supported calendars	Month	1 to 12	IN		Day	1 to 31	IN		Day of the week	ELIB_SCH_SUN (Sunday)	OUT		ELIB_SCH_MON (Monday)	ELIB_SCH_TUE (Tuesday)	ELIB_SCH_WED (Wednesday)	ELIB_SCH_THU (Thursday)	ELIB_SCH_FRI (Friday)	ELIB_SCH_SAT (Saturday)	Anniversary	0: Register new	IN/O	Any setting out of the scope is a
Data item name	Value	IN/O UT	Remark																																		
Application ID	Application ID	IN																																			
Year	Christian era	IN	Christian year in the supported calendars																																		
Month	1 to 12	IN																																			
Day	1 to 31	IN																																			
Day of the week	ELIB_SCH_SUN (Sunday)	OUT																																			
	ELIB_SCH_MON (Monday)																																				
	ELIB_SCH_TUE (Tuesday)																																				
	ELIB_SCH_WED (Wednesday)																																				
	ELIB_SCH_THU (Thursday)																																				
	ELIB_SCH_FRI (Friday)																																				
	ELIB_SCH_SAT (Saturday)																																				
Anniversary	0: Register new	IN/O	Any setting out of the scope is a																																		

			is assigned.
Repetition setting	ELIB_SCH_SINGLE (one-off)	IN	
	ELIB_SCH_YEAR (annual)		
Anniversary content	Text of anniversary content	IN	

- The day of the week is calculated by this function according to the date and set to the day of the week member in the structure delivered as an argument.
- The size of the anniversary content is 20 bytes.
(Schedule Service handles the anniversary content with the fixed size of 20 bytes, thus the size of the anniversary content size needs to be controlled by the user.)
- For a new anniversary registered successfully, the "anniversary number" is automatically set.
- After the "anniversary number" is automatically set, set it to the anniversary number member in the structure delivered as an argument.
- If there are anniversary data already registered to the date, the existing data of the anniversary is deleted. (See "G-8 Check another anniversary on identical date.")

Include file	srv_sch.h		
Prototype	Int Elib_SCH_Anniversary_Set(Ap_ID, AnniversaryData_p);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
AnniversaryData_p	ELIB_SCH_ANNIVERSARY_DATA *	I/O	Anniversary data structure
Return value	Type	I/O	Description
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Max registered Events reached : ELIB_SCH_MAXDATA
Remark			

Necessary procedure:

Create a MSB object by using MsbObjectCreate in respective applications.

5.62 Delete anniversary

Classification	Anniversary supporting function		
Description	Delete anniversary	Function name	Elib_SCH_Anniversary_Delete
Functional overview	<ul style="list-style-type: none"> - This function deletes an anniversary with a specified anniversary number or date. - This function sets the number or date of the anniversary to be deleted into the anniversary data structure represented by the anniversary data to be deleted. - If the anniversary data to be deleted has not been registered, this function returns with "Unregistered." - Specifying 0 to the anniversary number makes reference to the year, month, and day in the anniversary data structure and deletes the data registered to the date. - When a date is specified together with "registered number specified," the specified date is ignored. - For the date and time, set only "year, month, and day." 		
Include file	srv_sch.h		
Prototype	Int Elib_SCH_Anniversary_Delete(Ap_ID, AnniversaryData_p);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
AnniversaryData_p	_ELIB_SCH_ANNIVERSARY_DATA *	I	Anniversary data to be deleted
Return value	Type	I/O	Description
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered : ELIB_SCH_NODATA
Remark			
Necessary procedure: Create a MSB object by using MsbObjectCreate in respective applications.			

5.63 Delete all anniversaries

Classification	Anniversary supporting function		
Description	Delete all anniversaries	Function name	Elib_SCH_Anniversary_All_Delete
Functional overview	<p>- This function deletes all user-defined anniversaries.</p>		
Include file	srv_sch.h		
Prototype	Int Elib_SCH_Anniversary_All_Delete(Ap_ID);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Return value	Type	I/O	Description
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure: Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.64 Delete all anniversaries before specified date

Classification	Anniversary supporting function																	
Description	Delete all anniversaries before specified date	Function name	Elib_SCH_Anniversary_Past_Delete															
Functional overview	<div><ul style="list-style-type: none">- This function deletes all anniversaries registered before a specified date.- The application calls this function, setting the target date into the date of anniversary data structure represented by the date and time to be specified.- Target date and time (date of anniversary data structure)<table><tr><td>Data item name</td><td>Value</td><td>Remark</td></tr><tr><td>Year</td><td>Christian era</td><td>Christian year in the supported calendars</td></tr><tr><td>Month</td><td>1 to 12</td><td></td></tr><tr><td>Day</td><td>1 to 31</td><td></td></tr><tr><td>Day of the week</td><td></td><td>Unused</td></tr></table>- For any anniversary to be repeated (annually), its dates after the specified date of deletion are re-set.- For the scope of a date and time to be specified, see "6. Calendar Coverage."</div>			Data item name	Value	Remark	Year	Christian era	Christian year in the supported calendars	Month	1 to 12		Day	1 to 31		Day of the week		Unused
Data item name	Value	Remark																
Year	Christian era	Christian year in the supported calendars																
Month	1 to 12																	
Day	1 to 31																	
Day of the week		Unused																
Include file	srv_sch.h																	
Prototype	int Elib_SCH_Anniversary_Past_Delete(Ap_ID, date_p);																	
Argument	Type	I/O	Description															
Ap_ID	unsigned int	I	Application ID															
date_p	_ELIB_SCH_ANNIVERSARY_CAL_DATA *	I	Target date and time															
Return value	Type	I/O	Description															
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG															

		Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>	

5.65 Read anniversary

Classification	Anniversary supporting function		
Description	Read anniversary	Function name	Elib_SCH_Anniversary_Read
Functional overview	<ul style="list-style-type: none"> - This function sets the anniversary number into the anniversary data structure represented by the anniversary data readout area. The result is put into the said structure. - If the anniversary data to be read have not been registered, this function returns with "Unregistered." - Specifying 0 to the anniversary number makes reference to the year, month, and day in the anniversary data structure and reads the data registered to the date. - When a date is specified together with "registered number specified," the specified date is ignored. - The day of the week of data to be read is calculated by this function according to its date and set to the day of the week member in the structure delivered as an argument. - For the date and time, set only "year, month, and day." 		
Include file	srv_sch.h		
Prototype	Int Elib_SCH_Anniversary_Read(Ap_ID, AnniversaryData_p);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
AnniversaryData_p	ELIB_SCH_ANNIVERSARY_DATA *	I/O	Anniversary data readout area
Return value	Type	I/O	Description
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered : ELIB_SCH_NODATA
Remark	<p>Related message:</p> <p>None</p> <p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		



5.66 Read anniversary list

Classification	Anniversary supporting function																																								
Description	Read anniversary list	Function name	Elib_SCH_Anniversary_ListData_Read																																						
Functional overview	<ul style="list-style-type: none"> - This function sorts registered anniversaries in chronological order then reads them one by one. - The application calls this interface (function) once for each of anniversaries to be read. The readout data are put into the anniversary data structure represented by the anniversary data readout area. - Anniversary data readout area (anniversary data structure) <table> <tr> <th>Data item name</th><th>Value</th><th>IN/OUT</th><th>Remark</th></tr> <tr> <td>Application ID</td><td>Application ID</td><td>OUT</td><td></td></tr> <tr> <td>Year</td><td>Christian era</td><td>OUT</td><td>Christian year in the supported calendars</td></tr> <tr> <td>Month</td><td>1 to 12</td><td>OUT</td><td></td></tr> <tr> <td>Day</td><td>1 to 31</td><td>OUT</td><td></td></tr> <tr> <td rowspan="7">Day of the week</td><td>ELIB_SCH_SUN (Sunday)</td><td rowspan="7">OUT</td><td rowspan="7"></td></tr> <tr> <td>ELIB_SCH_MON (Monday)</td></tr> <tr> <td>ELIB_SCH_TUE (Tuesday)</td></tr> <tr> <td>ELIB_SCH_WED (Wednesday)</td></tr> <tr> <td>ELIB_SCH_THU (Thursday)</td></tr> <tr> <td>ELIB_SCH_FRI (Friday)</td></tr> <tr> <td>ELIB_SCH_SAT (Saturday)</td></tr> <tr> <td>Anniversary number</td><td>1 to 100</td><td>OUT</td><td></td></tr> <tr> <td>Repetition setting</td><td>ELIB_SCH_SINGLE (one-off)</td><td>OUT</td><td></td></tr> </table>			Data item name	Value	IN/OUT	Remark	Application ID	Application ID	OUT		Year	Christian era	OUT	Christian year in the supported calendars	Month	1 to 12	OUT		Day	1 to 31	OUT		Day of the week	ELIB_SCH_SUN (Sunday)	OUT		ELIB_SCH_MON (Monday)	ELIB_SCH_TUE (Tuesday)	ELIB_SCH_WED (Wednesday)	ELIB_SCH_THU (Thursday)	ELIB_SCH_FRI (Friday)	ELIB_SCH_SAT (Saturday)	Anniversary number	1 to 100	OUT		Repetition setting	ELIB_SCH_SINGLE (one-off)	OUT	
Data item name	Value	IN/OUT	Remark																																						
Application ID	Application ID	OUT																																							
Year	Christian era	OUT	Christian year in the supported calendars																																						
Month	1 to 12	OUT																																							
Day	1 to 31	OUT																																							
Day of the week	ELIB_SCH_SUN (Sunday)	OUT																																							
	ELIB_SCH_MON (Monday)																																								
	ELIB_SCH_TUE (Tuesday)																																								
	ELIB_SCH_WED (Wednesday)																																								
	ELIB_SCH_THU (Thursday)																																								
	ELIB_SCH_FRI (Friday)																																								
	ELIB_SCH_SAT (Saturday)																																								
Anniversary number	1 to 100	OUT																																							
Repetition setting	ELIB_SCH_SINGLE (one-off)	OUT																																							

	ELIB_SCH_YEAR (annual)		
Anniversary content	Text of anniversary content	OUT	

- The day of the week of data to be read is calculated by this function according to its date and set to the day of the week member in the structure delivered as an argument.
- An anniversary to be read includes repetitions of it.
- This function lists anniversaries at the initial call to it (made with 0 set to the listing number argument). In the second or later call, specifying the position of an anniversary in the listing number argument allows the anniversary data to be read. Note that the initial call returns the anniversary data at the top of the list.
- When the repetition setting is set to "annual," the nearest future day (date on which the event is repeated next) is set to the anniversary data structure and output. However, if the nearest future day is beyond the end date and time the calendar can support, an allowable latest day is set to the anniversary data structure and output.

Include file	srv_sch.h		
Prototype	int Elib_SCH_Anniversary_ListData_Read(Ap_ID, Anniversary_No, AnniversaryData_p);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Anniversary_No	int	I	Listing number
AnniversaryData_p	_ELIB_SCH_ANNI VERSARY_DATA *	O	Anniversary data readout area
Return value	Type	I/O	Description
ret	int	O	Successful : Number of events registered in the list Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered : ELIB_SCH_NODATA
Remark			

Necessary procedure:

Create a MSB object by using MsbObjectCreate in respective applications.

5.67 Read number of registered anniversaries

Classification	Anniversary supporting function		
Description	Read number of registered anniversaries	Function name	Elib_SCH_Anniversary_Total_Num_Read
Functional overview	<p>- This function gets the number of already registered anniversaries.</p>		
Include file	srv_sch.h		
Prototype	Int Elib_SCH_Anniversary_Total_Num_Read(Ap_ID);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Return value	Type	I/O	Description
ret	int	O	Successful : Number of anniversaries registered (0 to 100) Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	<p>Necessary procedure: Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.68 Check another anniversary on identical date

Classification	Anniversary supporting function																																	
Description	Check another anniversary on identical date	Function name	Elib_SCH_Anniversary_Same_Date_Check																															
Functional overview	<div><ul style="list-style-type: none">- This function checks whether there is another anniversary already registered to the date of a specified anniversary.- This function makes a check by using the "date specified, repetition setting, and registration number."- This function makes a check on combinations of repetition types as follows:</div> <table><tr><td>Holiday 1</td><td>Holiday 2</td><td>Condition 1</td><td>Condition 2</td><td>Result</td></tr><tr><td rowspan="6">One-off</td><td rowspan="2">One-off</td><td>Year, Month, Day ≠ Year, Month, Day</td><td>—</td><td>No other on the date</td></tr><tr><td>Year, Month, Day = Year, Month, Day</td><td>—</td><td>Another on the date</td></tr><tr><td rowspan="3">Annual</td><td>Month and Day of one-off ≠ Month and Day of annual</td><td>—</td><td>No other on the date</td></tr><tr><td rowspan="2">Month and Day of one-off = Month and Day of annual</td><td>Date (year) of one-off < Date (year) of annual</td><td>No other on the date</td></tr><tr><td>Date (year) of annual ≤ Date (year) of one-off</td><td>Another on the date</td></tr><tr><td rowspan="2">Annual</td><td rowspan="2">Annual</td><td>Month and Day ≠ Month and Day</td><td>—</td><td>No other on the date</td></tr><tr><td>Month and Day = Month and Day</td><td>—</td><td>Another on the date</td></tr></table>				Holiday 1	Holiday 2	Condition 1	Condition 2	Result	One-off	One-off	Year, Month, Day ≠ Year, Month, Day	—	No other on the date	Year, Month, Day = Year, Month, Day	—	Another on the date	Annual	Month and Day of one-off ≠ Month and Day of annual	—	No other on the date	Month and Day of one-off = Month and Day of annual	Date (year) of one-off < Date (year) of annual	No other on the date	Date (year) of annual ≤ Date (year) of one-off	Another on the date	Annual	Annual	Month and Day ≠ Month and Day	—	No other on the date	Month and Day = Month and Day	—	Another on the date
Holiday 1	Holiday 2	Condition 1	Condition 2	Result																														
One-off	One-off	Year, Month, Day ≠ Year, Month, Day	—	No other on the date																														
		Year, Month, Day = Year, Month, Day	—	Another on the date																														
	Annual	Month and Day of one-off ≠ Month and Day of annual	—	No other on the date																														
		Month and Day of one-off = Month and Day of annual	Date (year) of one-off < Date (year) of annual	No other on the date																														
			Date (year) of annual ≤ Date (year) of one-off	Another on the date																														
	Annual	Annual	Month and Day ≠ Month and Day	—	No other on the date																													
Month and Day = Month and Day			—	Another on the date																														
Include file	srv_sch.h																																	
Prototype	Int Elib_SCH_Anniversary_Same_Date_Check(Ap_ID, AnniversaryData_p);																																	
Argument	Type	I/O	Description																															
Ap_ID	unsigned int	I	Application ID																															
AnniversaryData_p	_ELIB_SCH_ANNIVERSARY_DATA *	I	Anniversary data																															
Return value	Type	I/O	Description																															
ret	Int	O	Registerable : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Another on the date : ELIB_SCH_SAMEDATE Incorrect parameter :																															

			ELIB_SCH_PARAM_NG
Remark			
<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>			

Return value	Type	I/O	Description
ret	int	O	Successful :Number of registered anniversaries Unsuccessful: ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered : ELIB_SCH_NODATA
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.70 Check for holiday/anniversary

Classification	Holiday/anniversary supporting functions																						
Description	Check for holiday/anniversary	Function name	Elib_SCH_HAM_DayPlan_Read																				
Functional overview	<p>- This function checks whether there are any holiday and/or anniversary registered to a specified target date.</p> <p>- Target date (date of holiday/anniversary/mydiary data structure)</p> <table> <tr> <th>Data item name</th><th>Value</th><th>IN/OUT</th><th>Remark</th></tr> <tr> <td>Year</td><td>Christian era</td><td>IN</td><td>Christian year in the supported calendars</td></tr> <tr> <td>Month</td><td>1 to 12</td><td>IN</td><td></td></tr> <tr> <td>Day</td><td>1 to 31</td><td>IN</td><td></td></tr> <tr> <td>Day of the week</td><td>0 to 6 (corresponding to Sun to Sat)</td><td>OUT</td><td></td></tr> </table> <p>- When a holiday and/or anniversary has been set, a logical sum is taken for the following values and returned (0 for an unspecified item):</p> <pre> ELIB_SCH_HOLIDAY_EXIST /* Holiday */ ELIB_SCH_DEFAULT_HOLIDAY_EXIST /* Default holiday */ ELIB_SCH_ANNIVERSARY_EXIST /* Anniversary */ </pre> <p>- The day of the week of the target date is calculated by this function according to its date and set to the day of the week member in the structure delivered as an argument.</p>			Data item name	Value	IN/OUT	Remark	Year	Christian era	IN	Christian year in the supported calendars	Month	1 to 12	IN		Day	1 to 31	IN		Day of the week	0 to 6 (corresponding to Sun to Sat)	OUT	
Data item name	Value	IN/OUT	Remark																				
Year	Christian era	IN	Christian year in the supported calendars																				
Month	1 to 12	IN																					
Day	1 to 31	IN																					
Day of the week	0 to 6 (corresponding to Sun to Sat)	OUT																					
Include file	srv_sch.h																						
Prototype	Int Elib_SCH_HAM_DayPlan_Read(Ap_ID, date_p);																						
Argument	Type	I/O	Description																				
Ap_ID	unsigned int	I	Application ID																				
date_p	_ELIB_SCH_HAM_CAL_DATA *	I/O	Target date																				
Return value	Type	I/O	Description																				

ret	Int	O	Successful : Setting status Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered : ELIB_SCH_NODATA
Remark	<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>		

for an unspecified item):

ELIB_SCH_HOLIDAY_EXIST	/* Holiday	*/
ELIB_SCH_DEFAULT_HOLIDAY_EXIST	/* Default holiday	*/
ELIB_SCH_ANNIVERSARY_EXIST	/* Anniversary	*/

- The value of the target year and month is to be checked for the scope. For the scope, see "6. Calendar Coverage."

Include file	srv_sch.h		
Prototype	int Elib_SCH_HAM_DayPlan_Read_Month(Ap_ID, date_p, info);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
date_p	_ELIB_SCH_HAM_CAL_DATA *	I	Target year and month
info	unsigned char *	O	Holiday information storage array
Return value	Type	I/O	Description
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered : ELIB_SCH_NODATA
Remark			
<p>Necessary procedure:</p> <p>Create a MSB object by using MsbObjectCreate in respective applications.</p>			

5.72 Register ADL reserved time event

Classification	ADL reservation update supporting function																																												
Description	Register ADL reserved time event	Function name	Elib_SCH_ADL_Time_Set																																										
Functional overview	<ul style="list-style-type: none"> - This function registers the ADL reserved time. - The number of ADL reserved time registerable must be 1. - If there is any ADL reserved time that has been already registered, this function overwrites it with the specified date and time. - This function sets the data to be registered into the ADL reserved time data structure represented by the ADL reserved time data to be registered. (For details about the structure, see "4. List of Structures.") - ADL reserved time data to be registered (ADL reserved time data structure) <table> <tr> <th>Data item name</th><th>Value</th><th>IN/OUT</th><th>Remark</th></tr> <tr> <td>Application ID</td><td>Application ID</td><td>IN</td><td></td></tr> <tr> <td>Year</td><td>Christian era</td><td>IN</td><td>Christian year in the supported calendars</td></tr> <tr> <td>Month</td><td>1 to 12</td><td>IN</td><td></td></tr> <tr> <td>Day</td><td>1 to 31</td><td>IN</td><td></td></tr> <tr> <td>Hour</td><td>0 to 23</td><td>IN</td><td></td></tr> <tr> <td>Minute</td><td>0 to 59</td><td>IN</td><td></td></tr> <tr> <td>Second</td><td>0 to 59</td><td>IN</td><td>Fixed to 0</td></tr> <tr> <td rowspan="7">Day of the week</td><td>ELIB_SCH_SUN (Sunday)</td><td rowspan="7">IN</td><td rowspan="7">Unused</td></tr> <tr> <td>ELIB_SCH_MON (Monday)</td></tr> <tr> <td>ELIB_SCH_TUE (Tuesday)</td></tr> <tr> <td>ELIB_SCH_WED (Wednesday)</td></tr> <tr> <td>ELIB_SCH_THU (Thursday)</td></tr> <tr> <td>ELIB_SCH_FRI (Friday)</td></tr> <tr> <td>ELIB_SCH_SAT (Saturday)</td></tr> </table> <ul style="list-style-type: none"> - To receive notifications about an event specified by this interface (function), the application needs to make a call to the "A-19 Start posting alarm notifications" function when it starts up to inform 			Data item name	Value	IN/OUT	Remark	Application ID	Application ID	IN		Year	Christian era	IN	Christian year in the supported calendars	Month	1 to 12	IN		Day	1 to 31	IN		Hour	0 to 23	IN		Minute	0 to 59	IN		Second	0 to 59	IN	Fixed to 0	Day of the week	ELIB_SCH_SUN (Sunday)	IN	Unused	ELIB_SCH_MON (Monday)	ELIB_SCH_TUE (Tuesday)	ELIB_SCH_WED (Wednesday)	ELIB_SCH_THU (Thursday)	ELIB_SCH_FRI (Friday)	ELIB_SCH_SAT (Saturday)
Data item name	Value	IN/OUT	Remark																																										
Application ID	Application ID	IN																																											
Year	Christian era	IN	Christian year in the supported calendars																																										
Month	1 to 12	IN																																											
Day	1 to 31	IN																																											
Hour	0 to 23	IN																																											
Minute	0 to 59	IN																																											
Second	0 to 59	IN	Fixed to 0																																										
Day of the week	ELIB_SCH_SUN (Sunday)	IN	Unused																																										
	ELIB_SCH_MON (Monday)																																												
	ELIB_SCH_TUE (Tuesday)																																												
	ELIB_SCH_WED (Wednesday)																																												
	ELIB_SCH_THU (Thursday)																																												
	ELIB_SCH_FRI (Friday)																																												
	ELIB_SCH_SAT (Saturday)																																												

that the application is ready to receive event notifications.

- For the scope of the year, month, and day, see "6. Calendar Coverage."

Include file	srv_sch.h		
Prototype	Int Elib_SCH_ADL_Time_Set(Ap_ID, data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
data	_ELIB_SCH_ADL_DATA *	I	ADL reserved time data to be registered
Return value	Type	-	
ret	Int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark			
Necessary procedure: Create a MSB object by using MsbObjectCreate in respective applications.			

5.73 Delete ADL reserved time event

Classification	ADL reservation update supporting function		
Description	Delete ADL reserved time event	Function name	Elib_SCH_ADL_Time_Reset
Functional overview	<p>- This function deletes the ADL reserved time registered by using "I-1 Register ADL reserved time event."</p>		
Include file	srv_sch.h		
Prototype	Int Elib_SCH_ADL_Time_Reset(Ap_ID);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Return value	Type	-	
ret	Int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered : ELIB_SCH_NODATA
Remark	<p>Necessary procedure: Create a MSB object by using MsbObjectCreate in respective applications.</p>		

5.74 Read ADL reserved time event

Classification	ADL reservation update supporting function																																												
Description	Read ADL reserved time event	Function name	Elib_SCH_ADL_Time_Read																																										
Functional overview	<div><div>- This function reads the ADL reserved time registered into the ADL reserved time data structure represented by the ADL reserved time data readout area. (For details about the structure, see "4. List of Structures.")</div><div>- ADL reserved time data readout area (ADL reserved time data structure)</div></div> <table><tr><td>Data item name</td><td>Value</td><td>IN/OUT</td><td>Remark</td></tr><tr><td>Application ID</td><td>Application ID</td><td>OUT</td><td></td></tr><tr><td>Year</td><td>Christian era</td><td>OUT</td><td>Christian year in the supported calendars</td></tr><tr><td>Month</td><td>1 to 12</td><td>OUT</td><td></td></tr><tr><td>Day</td><td>1 to 31</td><td>OUT</td><td></td></tr><tr><td>Hour</td><td>0 to 23</td><td>OUT</td><td></td></tr><tr><td>Minute</td><td>0 to 59</td><td>OUT</td><td></td></tr><tr><td>Second</td><td>0 to 59</td><td>OUT</td><td></td></tr><tr><td rowspan="7">Day of the week</td><td>ELIB_SCH_SUN (Sunday)</td><td rowspan="7">OUT</td><td rowspan="7"></td></tr><tr><td>ELIB_SCH_MON (Monday)</td></tr><tr><td>ELIB_SCH_TUE (Tuesday)</td></tr><tr><td>ELIB_SCH_WED (Wednesday)</td></tr><tr><td>ELIB_SCH_THU (Thursday)</td></tr><tr><td>ELIB_SCH_FRI (Friday)</td></tr><tr><td>ELIB_SCH_SAT (Saturday)</td></tr></table>			Data item name	Value	IN/OUT	Remark	Application ID	Application ID	OUT		Year	Christian era	OUT	Christian year in the supported calendars	Month	1 to 12	OUT		Day	1 to 31	OUT		Hour	0 to 23	OUT		Minute	0 to 59	OUT		Second	0 to 59	OUT		Day of the week	ELIB_SCH_SUN (Sunday)	OUT		ELIB_SCH_MON (Monday)	ELIB_SCH_TUE (Tuesday)	ELIB_SCH_WED (Wednesday)	ELIB_SCH_THU (Thursday)	ELIB_SCH_FRI (Friday)	ELIB_SCH_SAT (Saturday)
Data item name	Value	IN/OUT	Remark																																										
Application ID	Application ID	OUT																																											
Year	Christian era	OUT	Christian year in the supported calendars																																										
Month	1 to 12	OUT																																											
Day	1 to 31	OUT																																											
Hour	0 to 23	OUT																																											
Minute	0 to 59	OUT																																											
Second	0 to 59	OUT																																											
Day of the week	ELIB_SCH_SUN (Sunday)	OUT																																											
	ELIB_SCH_MON (Monday)																																												
	ELIB_SCH_TUE (Tuesday)																																												
	ELIB_SCH_WED (Wednesday)																																												
	ELIB_SCH_THU (Thursday)																																												
	ELIB_SCH_FRI (Friday)																																												
	ELIB_SCH_SAT (Saturday)																																												
Include file	srv_sch.h																																												
Prototype	Int Elib_SCH_ADL_Time_Read(Ap_ID, data);																																												

Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
data	_ELIB_SCH_ADL_DATA *	O	ADL reserved time data readout area
Return value	Type	-	
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG Unregistered : ELIB_SCH_NODATA
Remark			

Necessary procedure:

Create a MSB object by using MsbObjectCreate in respective applications. (For details, refer to "MSB External API Specifications.")

5.75 Read ADL re-calculated time

Classification	ADL reservation update supporting function																																		
Description	Read ADL re-calculated time	Function name	Elib_SCH_ADL_Calculation_Time_Read																																
Functional overview	<ul style="list-style-type: none"> - This function reads data for ADL re-calculated time. - Before-change/after-change system time readout area (date and time data structure) <table border="1"> <thead> <tr> <th>Data item name</th><th>Value</th><th>IN/OUT</th><th>Remark</th></tr> </thead> <tbody> <tr> <td>Year</td><td>Christian era</td><td>OUT</td><td>Christian year in the supported calendars</td></tr> <tr> <td>Month</td><td>1 to 12</td><td>OUT</td><td></td></tr> <tr> <td>Day</td><td>1 to 31</td><td>OUT</td><td></td></tr> <tr> <td>Day of the week</td><td>0 to 6 (corresponding to Sun to Sat)</td><td>OUT</td><td></td></tr> <tr> <td>Hour</td><td>0 to 23</td><td>OUT</td><td></td></tr> <tr> <td>Minute</td><td>0 to 59</td><td>OUT</td><td></td></tr> <tr> <td>Second</td><td>0 to 59</td><td>OUT</td><td></td></tr> </tbody> </table> <ul style="list-style-type: none"> - The data for ADL re-calculated time indicates the before-change system time and the after-change system time. - The before-change system time and after-change system time are initialized to 0 when the power is turned on and no record is kept about changes for the system time. 			Data item name	Value	IN/OUT	Remark	Year	Christian era	OUT	Christian year in the supported calendars	Month	1 to 12	OUT		Day	1 to 31	OUT		Day of the week	0 to 6 (corresponding to Sun to Sat)	OUT		Hour	0 to 23	OUT		Minute	0 to 59	OUT		Second	0 to 59	OUT	
Data item name	Value	IN/OUT	Remark																																
Year	Christian era	OUT	Christian year in the supported calendars																																
Month	1 to 12	OUT																																	
Day	1 to 31	OUT																																	
Day of the week	0 to 6 (corresponding to Sun to Sat)	OUT																																	
Hour	0 to 23	OUT																																	
Minute	0 to 59	OUT																																	
Second	0 to 59	OUT																																	
Include file	srv_sch.h																																		
Prototype	int Elib_SCH_ADL_Calculation_Time_Read (Ap_ID, before_time_p, after_time_p);																																		
Argument	Type	I/O	Description																																
Ap_ID	unsigned int	I	Application ID																																
Before_time_p	ELIB_SCH_CAL_DATA *	O	Before-change system time readout area																																
after_time_p	ELIB_SCH_CAL_DATA *	O	After-change system time readout area																																

	*		
Return value	Type	I/O	
ret	int	O	Successful : ELIB_SCH_OK Unsuccessful : ELIB_SCH_NG Incorrect parameter : ELIB_SCH_PARAM_NG
Remark	Necessary procedure: Create a MSB object by using MsbObjectCreate in respective applications.		

6. Data Exchange Service

6.1 Start Event notification request

Classification	Data exchange service support function		
Function	Start Event notification request	Symbol	Elib_DEX_Request
Functional overview	<p>- Registers notification requests of events provided by the data exchange service for the application.</p> <p>- This function is called only for the notification-required events. Event notification can be registered for registration functions (telephone directly/SMS/certificate DL client authentication).</p> <p>- When a notification request is registered, the event is posted. The API can accept and process events as required.</p> <p>- If a notification event is no longer required, it can be cleared as described in Stop Event Notification Request.</p> <p>- Intermittent power loss or turning the power supply off disables the event notification requests set using this interface (function).</p>		
Include file	srv_dex.h msb/msb.h (Provided from msb)		
Calling sequence	int Elib_DEX_Request(ap_id, event, func);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
event	int	I	Notification event
func	MsbFunc	I	Pointer to the function that executes callback when an event occurs (*1)
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error If an error occurs at each function registration request, registration will partially fail.
Remark	<p>*1: The type of function to be called back when an event occurs is shown below.</p> <pre>void func(MsbObject *client, MsbObject *server, void *sendData, void *recvDate, void *data, MsbEnvironment *ev);</pre> <p>MSB is used for event notification.</p>		

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate.

Note

6.2 Stop Event notification request

Classification	Data exchange service support function		
Function	Stop Event notification request	Symbol	Elib_DEX_Cancel
Functional overview	<p>- Stops event notification registered as described in Start Event Notification Request.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Cancel(ap_id, event);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
event	Int	I	Type of event for which notification is to be stopped
Return value	Type	I/O	Description
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error If an error occurs at each function stop request, stop will partially fail.
Remark	<p>Necessary procedure - For each application, create a MSB object in advance using MsbObjectCreate.</p> <p>Note</p>		

6.3 Language List Collection

Classification	Data exchange service support function																				
Function	Language list request	Symbol	Elib_DEX_LanguageListReq																		
Functional overview	<div>- Requests language information list. -> Returns the language list stored in the UIM.</div> <div>- When the power supply is turned on, the language is also posted by the event below.</div>																				
Include file	srv_dex.h																				
Calling sequence	int Elib_DEX_LanguageListReq(ap_id);																				
Argument	Type	I/O	Description																		
ap_id	unsigned int	I	Application ID																		
Return value	Type	I/O	Description																		
Ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error If an error occurs or a parameter is incorrect, event notification will not be performed.																		
Remark	<div>- This interface is used to request collection of the language list information in the UIM. The list information is posted by the event below after the request.</div> <div>Table Language list information notification event format (_ELIB_DEX_EVT_LANG)</div> <table><tr><th>Member name</th><th>Type</th><th>Description</th></tr><tr><td>category</td><td>Int</td><td>DataExchangeNotify</td></tr><tr><td>subtype</td><td>Int</td><td>DataExcNotify_LanguageListRes (Posts the language list information.)</td></tr><tr><td>info</td><td>Int</td><td>Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure</td></tr><tr><td>subinfo</td><td>Int</td><td>List count (0 to 5)</td></tr><tr><td>data</td><td>ELIB_LANG_INF</td><td>Language information structure (*1)</td></tr></table> <div>If the processing result is failure, the subinfo/data is invalid (NULL).</div> <div>*1: Language information structure typedef struct _ELIB_LANG_INF { int code ; /* Language code char lang_name[LANG_NAME_MAX] ; /* List display character string char lang_flg ; /* Available flag</div>			Member name	Type	Description	category	Int	DataExchangeNotify	subtype	Int	DataExcNotify_LanguageListRes (Posts the language list information.)	info	Int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure	subinfo	Int	List count (0 to 5)	data	ELIB_LANG_INF	Language information structure (*1)
Member name	Type	Description																			
category	Int	DataExchangeNotify																			
subtype	Int	DataExcNotify_LanguageListRes (Posts the language list information.)																			
info	Int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure																			
subinfo	Int	List count (0 to 5)																			
data	ELIB_LANG_INF	Language information structure (*1)																			

```

} _ELIB_LANG_INF ;
#define LANG_NAME_MAX      15          /* Maximum length */

```

Table Supported language lists (Up to five information items are passed.)

Language code	List display character string	Available flag
ELIB_DEX_JAPANESE	Japanese	ON
ELIB_DEX_ENGLISH	English	ON

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

6.4 Memory Language Change Request

Classification	Data exchange service support function		
Function	Memory language change request	Symbol	Elib_DEX_MEMSetLanguageReq
Functional overview	<p>- Requests language setting modification on SRAM. -> Switches the language setting stored in SRAM.</p> <p>The result of this interface is not an event notification.</p>		
srv_dex.h	srv_dex.h		
Calling sequence	int Elib_DEX_MEMSetLanguageReq(ap_id, language);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
language	int	I	Language code (See Table 3.B-1.2.)
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark			

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..

Note

6.5 USIM Language Setting Request

Classification	Data exchange service support function																				
Function	USIM language setting request	Symbol	Elib_DEX_SetLanguageReq																		
Functional overview	<div>- Requests language setting modification in the USIM.</div> <div>-> Switches the language setting stored in the USIM.</div>																				
Include file	srv_dex.h																				
Calling sequence	int Elib_DEX_SetLanguageReq(ap_id, language);																				
Argument	Type	I/O	Description																		
ap_id	unsigned int	I	Application ID																		
language	int	I	Language code																		
Return value	Type	I/O	Description																		
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error If an error occurs or a parameter is incorrect, event notification will not be performed.																		
Remark	<div>- The language setting is requested using this interface and the result is posted by the event below after the request.</div> <div>Table Language switching notification event format (_ELIB_DEX_EVENT)</div> <table><tr><th>Member name</th><th>Type</th><th>Description</th></tr><tr><td>category</td><td>int</td><td>DataExchangeNotify</td></tr><tr><td>subtype</td><td>int</td><td>DataExcNotify_SetLanguageRes (Notification result of language switching request)</td></tr><tr><td>Info</td><td>int</td><td>Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure</td></tr><tr><td>subinfo</td><td>int</td><td>Unused (always NULL)</td></tr><tr><td>data</td><td>int</td><td>Unused (always NULL)</td></tr></table> <div>Necessary procedure<div>- For each application, create a MSB object in advance using MsbObjectCreate..</div><div>- Client needs to be registered at the event (see [Start Event Notification Request]).</div></div> <div>Note</div>			Member name	Type	Description	category	int	DataExchangeNotify	subtype	int	DataExcNotify_SetLanguageRes (Notification result of language switching request)	Info	int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure	subinfo	int	Unused (always NULL)	data	int	Unused (always NULL)
Member name	Type	Description																			
category	int	DataExchangeNotify																			
subtype	int	DataExcNotify_SetLanguageRes (Notification result of language switching request)																			
Info	int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure																			
subinfo	int	Unused (always NULL)																			
data	int	Unused (always NULL)																			

6.6 Get current language request

Classification	Data exchange service support function		
Function	Get current language request	Symbol	Elib_DEX_RefLanguage
Functional overview	<p>- Inquires about the language currently set. -> Returns the set language stored in SRAM.</p> <p>The result of this interface is not an event notification.</p>		
Include file	srv_dex.h		
Calling sequence	Int Elib_DEX_RefLanguage(void);		
Argument	Type	I/O	Description
None		I	
Return value	Type	I/O	Description
language	int	O	Processing result Language code: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark	<p>Necessary procedure - For each application, create a MSB object in advance using MsbObjectCreate..</p> <p>Note</p>		

6.7 Get Self Phone Number (MSISDN) Request

Classification	Data exchange service support function																				
Function	Get self phone number (MSISDN) request	Symbol	Elib_DEX_MSISDNRefReq																		
Functional overview	<div>- Requests access of the self phone number.</div>																				
Include file	srv_dex.h																				
Calling sequence	int Elib_DEX_MSISDNRefReq(ap_id);																				
Argument	Type	I/O	Description																		
ap_id	unsigned int	I	Application ID																		
Return value	Type	I/O	Description																		
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error If an error occurs or a parameter is incorrect, event notification will not be performed.																		
Remark	<div>- This interface is used to request collection of the local station number (MSISDN). The result is posted by the event below after the request.</div> <div>Table MSISDN access request result notification event format (_ELIB_DEX_EVT_MSISDN)</div> <table><tr><th>Member name</th><th>Type</th><th>Description</th></tr><tr><td>category</td><td>int</td><td>DataExchangeNotify</td></tr><tr><td>subtype</td><td>int</td><td>DataExcNotify_MSISDNRefRes (Posts the result of the MSISDN access request.)</td></tr><tr><td>info</td><td>int</td><td>Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure</td></tr><tr><td>subinfo</td><td>int</td><td>Unused (always NULL)</td></tr><tr><td>data</td><td>ELIB_TDIAL_INF</td><td>Dial information structure (*1)</td></tr></table> <div>If the processing result is failure, subinfo/data is invalid (NULL).</div> <div>*1: Dial information structure</div> <pre>typedef struct _ELIB_TDIAL_INF { unsigned char len ; /* Dial length */ char dial[ELIB_MSISDN_DIAL_MAX] ; /* Dial */ } _ELIB_TDIAL_INF ; #define ELIB_MSISDN_DIAL_MAX 26</pre>			Member name	Type	Description	category	int	DataExchangeNotify	subtype	int	DataExcNotify_MSISDNRefRes (Posts the result of the MSISDN access request.)	info	int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure	subinfo	int	Unused (always NULL)	data	ELIB_TDIAL_INF	Dial information structure (*1)
Member name				Type	Description																
category	int	DataExchangeNotify																			
subtype	int	DataExcNotify_MSISDNRefRes (Posts the result of the MSISDN access request.)																			
info	int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure																			
subinfo	int	Unused (always NULL)																			
data	ELIB_TDIAL_INF	Dial information structure (*1)																			

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

6.8 Time/charges Phone Call Reset Request

Classification	Data exchange service support function											
Function	Time/charges phone call reset request	Symbol	Elib_DEX_DCMFrqClrReq									
Functional overview	<div>- Requests resetting (zero clear) of the conversation items.<div>- All items</div><div>- Conversation time</div><div>- Additional conversation time...Per call type (reset regardless of voice or digital communication.)</div></div>											
Include file	srv_dex.h											
Calling sequence	int Elib_DEX_DCMFrqClrReq(ap_id , kind);											
Argument	Type	I/O	Description									
ap_id	unsigned int	I	Application ID									
kind	int	I	Initialization item specification ELIB_DEX_TKALL : Initialize all items. ELIB_DEX_TKALLTIME : Initialize conversation time/total time. ELIB_DEX_TKTIME : Initialize conversation time. ELIB_DEX_TKTTLTIME : Initialize total time.									
Return value	Type	I/O	Description									
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error If an error occurs or a parameter is incorrect, event notification will not be performed.									
Remark												
<div>- This interface is used to request resetting of the total time in the USIM. The result is posted by the event below after the request.</div> <div>Table Total time reset request result notification event format (_ELIB_DEX_EVENT)</div> <table><tr><th>Member name</th><th>Type</th><th>Description</th></tr><tr><td>category</td><td>int</td><td>DataExchangeNotify</td></tr><tr><td>subtype</td><td>int</td><td>DataExcNotify_DCMFrqClrRes (Posts the result of the total time reset request.)</td></tr></table>				Member name	Type	Description	category	int	DataExchangeNotify	subtype	int	DataExcNotify_DCMFrqClrRes (Posts the result of the total time reset request.)
Member name	Type	Description										
category	int	DataExchangeNotify										
subtype	int	DataExcNotify_DCMFrqClrRes (Posts the result of the total time reset request.)										

Info	int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure
subinfo	int	Unused (always NULL)
data	int	Unused (always NULL)

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

6.9 Regular Reception Start Request

Classification	Data exchange service support function		
Function	Regular reception start request	Symbol	Elib_DEX_SrvStartInd
Functional overview	<p>- Requests regular reception (incoming enabled status) start.</p> <p>- Even if PIN1 input is set to off, reception is handled as authentication completed when PIN1 unnecessary notification is received from the ELIB authentication service. This interface must be executed before transiting to wait status.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_SrvStartInd (pin_sts) ;		
Argument	Type	I/O	Description
pin_sts	int	I	Wait transition cause ELIB_DEX_PIN1 : PIN1 completion ELIB_DEX_PUK : PUK completion
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark	<p>- After PIN authentication is completed (after the processing described above) at activation, this interface must be called before transiting to wait status. ELIB wait transition is executed when this interface requests regular acceptance start.</p> <p>- The causes when transiting to wait status are as follows: 1) After PIN1 authentication is completed when PIN1 is set to on. 2) After accepting a PN1 unnecessary response by the authentication application (PIN1 completion handling) when PIN1 is set to off. 3) After PUK authentication is completed when PUK is input at PIN1 failure.</p> <p>!The processing result is only a return value. There is no response event.</p> <p>Necessary procedure</p> <p>- For each application, create a MSB object in advance using MsbObjectCreate..</p> <p>Note</p>		

6.10 USIM version number request

Classification	Data exchange service support function		
Function	USIM version number request	Symbol	Elib_DEX_GetVerNo
Functional overview	<p>- Returns the USIM card version number.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_GetVerNo (ap_id, ver_no) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
ver_no	Unsigned char *	O	UIM version number Valid only when the return value is normal.
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark	<p>- This interface is only valid after PIN1 authentication. That is, after the status transits to regular wait status (incoming enabled). If this interface is called before this, ELIB_DEX_NG will be returned.</p> <p>- If EF_UICC (file for which a version number is defined) cannot be accessed, ELIB_DEX_NG will be returned.</p> <p>Necessary procedure</p> <p>- For each application, create a MSB object in advance using MsbObjectCreate..</p> <p>Note</p>		

6.11 Conversation/Total Time Request

Classification	Data exchange service support function		
Function	Conversation/total time request	Symbol	Elib_DEX_ComTalkTimeRef
Functional overview	<p>- Returns the previous conversation type and conversation time and each total time.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_ComTalkTimeRef(talk_time);		
Argument	Type	I/O	Description
talk_time	_ELIB_COMTALKTIME *	O	Pointer to the conversation/total time (*1)
Return value	Type	I/O	Description
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMEERR: Parameter error
Remark	<p>*1: Conversation/total time structure</p> <pre>typedef struct _ELIB_COMTALKTIME { int lastKind ; /* Previous conversation type (The following definition values are returned.) */ /* ELIB_TALK_NONE No conversation (initial value) */ /* ELIB_TALK_FOMA FOMA conversation */ /* ELIB_TALK_TVPHONE Videophone conversation */ /* ELIB_TALK_DIGITAL Digital communication */ int lastTime ; /* Previous conversation time */ int totalTime_FOMA ; /* Total time (FOMA conversation) */ int totalTime_VoIP ; /* Unused */ int totalTime_TVPhone ; /* Total time (Videophone conversation) */ int totalTime_Digital ; /* Total time (Digital communication) */ }_ELIB_COMTALKTIME ;</pre> <p>- This interface is used to collect the conversation type, conversation time, and each total time.</p>		

Return value	Type	I/O	Description
Ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_NOTSET: Not set (There is no data for the specified memory number.) ELIB_DEX_PRMERR: Parameter error
Remark	<pre> [Telephone directory data structure] typedef struct _ELIB_MEDDATA { unsigned short mem_no ; /* Memory number */ unsigned short address_entry ; /* Address entry 0: Not registered, Icon number: Registered */ unsigned short memo_entry ; /* Memo entry 0: Not registered, Icon number: Registered */ unsigned short image_entry ; /* Image entry 0: Not registered, Icon number: Registered */ unsigned short movie_entry ; /* Movie entry 0: Not registered, Icon number: Registered */ unsigned short dial_cnt ; /* Telephone number count */ unsigned short mail_cnt ; /* Mail address count */ unsigned short sip_cnt ; /* Unused */ unsigned short url_imode_cnt ; /* URL count */ unsigned short url_wlan_cnt ; /* Unused */ _ELIB_MEDDIAL dial[ELIB_DIAL_MAX] ; /* Telephone number data structure *4 */ _ELIB_MEDMAIL mail[ELIB_MAIL_MAX] ; /* Mail address structure *5 */ _ELIB_MEDSIP sip[ELIB_SIP_MAX] ; /*Unused */ _ELIB_MEDURL url_imode[ELIB_URL_MAX] ; /* URL data structure *6 */ _ELIB_MEDURL url_wlan[ELIB_URL_MAX] ; /*Unused *6 */ unsigned char kana_sei[ELIB_KANAMAX + 1] ; /* Read kana (family name) */ unsigned char kana_name[ELIB_KANAMAX + 1] ; /* Read kana (name) Body only */ unsigned char kanji_sei[ELIB_KANJIMAX + 1] ; /* Name (family name) */ unsigned char kanji_name[ELIB_KANJIMAX + 1] ; /* Name (name) Body only */ unsigned char secret ; /* Secret information Body only *7 */ unsigned char group ; /* Group number */ } _ELIB_MEDDATA ; #define ELIB_DIAL_MAX : 4 /* Telephone number maximum count */ ..For UIM, 1 #define ELIB_MAIL_MAX : 3 /* Mail address maximum count */ ..For UIM, 1 #define ELIB_URL_MAX : 2 /* URL maximum count */ ..For UIM, 0 #define ELIB_KANAMAX :32 /* Read kana */ ..For UIM, 25 bytes are valid. #define ELIB_KANJIMAX :32 /* Name */ ..For UIM, 21 bytes are valid. #define ELIB_DIAL_MAX : 4 /* Telephone number maximum count */ ..For UIM, only the leading data (first item) is valid. #define ELIB_MAIL_MAX : 3 /* Mail address maximum count */ ..For UIM, only the </pre>		

leading data (first item) is valid.

- NULL terminating character string for the read kana (family name/name), name (family name/name), telephone number, and mail address.
- For the name data and read kana data, the joined family name and name are valid only up to **32 bytes**. If the total of the family name and name exceeds **32 bytes**, the excess amount will be truncated from the name data.
- For UIM telephone directory data, only the family name data is valid. **The valid size of the UIM family name data is name: 21 bytes and read kana: 25 bytes.**

(*4) Telephone number data structure

```
typedef struct _ELIB_MEDDIAL
```

```
{
    unsigned char    dial[ ELIB_DIALMAX + 1 ] ; /* Telephone number          *4-1 */
    unsigned short   icon ; /* Icon information    Body only    *4-2 */
    unsigned char    scode[ ELIB_PINMAX + 1 ] ; /* Secret code        Body only    */
    unsigned char    kind ; /* Number type/telephone number identifier*4-3 */
    unsigned char    mode ; /* Issuing mode          *4-4 */
} _ELIB_MEDDIAL ;
```

```
#define ELIB_DIALMAX      26 /* Telephone number maximum number of digits */
```

```
#define ELIB_PINMAX      4 /* Secret code number of digits */
```

(*4-1) For UIM, the following setting value.

```
#define ELIB_UIM_DIALMAX  26 /* UIM telephone number maximum number of digits */
```

(*4-2) Icon types set for the icon information

```
ELIB_DEX_LICON_TELNO /* Telephone number */
ELIB_DEX_LICON_PDC /* Portable telephone */
ELIB_DEX_LICON_TVPHONE /* Videophone */
ELIB_DEX_LICON_HOUSE /* Home */
ELIB_DEX_LICON_COMPANY /* Company */
ELIB_DEX_LICON_COMPFAX /* Company FAX */
ELIB_DEX_LICON_SHOP /* Shop */
ELIB_DEX_LICON_FACTORY /* Factory */
ELIB_DEX_LICON_REPRESENT /* Representative */
ELIB_DEX_LICON_DIRECT /* Direct */
ELIB_DEX_LICON_SECRET /* Secret */
ELIB_DEX_LICON_HOTEL /* Hotel */
ELIB_DEX_LICON_SCHOOL /* School */
ELIB_DEX_LICON_HOMEFAX /* Home FAX */
ELIB_DEX_LICON_COMPUTER /* Personal computer */
ELIB_DEX_LICON_PAGER /* Pager */
ELIB_DEX_LICON_FAMILY /* Family */
ELIB_DEX_LICON_WOMAN /* Woman friend */
ELIB_DEX_LICON_MAN /* Male friend */
ELIB_DEX_LICON_FRIEND /* Friend */
ELIB_DEX_LICON_PHS /* PHS */
ELIB_DEX_LICON_HOME /* Home */
ELIB_DEX_LICON_RESTAURANT /* Restaurant */
ELIB_DEX_LICON_HOSPITAL /* Hospital */
```



```

(*6) URL data structure
typedef struct _ELIB_MEDURL
{
    unsigned char        url[ ELIB_URLMAX + 1 ] ; /* URL */
    unsigned short       icon ; /* Icon information Body only *4-2 */
} _ELIB_MEDURL ;

#define ELIB_URLMAX      256 /* URL maximum number of digits */

(*7) Secret information symbol
#define ELIB_SECRET_OFF      100 /* Not secre */
#define ELIB_SECRET_ON      101 /* Secret */

```

Necessary procedure

- For each application, a MSB object must be created in advance using MsbObjectCreate.

Note

6.13 Memory Dial Additional Information Request

Classification	Data exchange service support function		
Function	Memory dial additional information request	Symbol	Elib_DEX_Mem_DialOptionRead
Functional overview	<p>- Returns the additional information (address/memo/image/movie) registered in the telephone directory data of the specified memory number.</p> <p>- Checks the registration status of the additional information using the additional information entry value (0: No registration/Icon number: Registration) of the telephone directory data structure and uses this interface to collect the data if data collection is required.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Mem_DialOptionRead (ap_id, mem_no, op_info) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
mem_no	unsigned short	I	Body memory number (0 to 699) * Only the body telephone number data is valid.
Op_info	_ELIB_MEDOP_INFO	I/O	Additional information structure (See the Functional description.)
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_NOENTRY: Not registered (*1) ELIB_DEX_PRMERR: Parameter error
Remark			

[Memory dial additional information structure]

```
typedef struct _ELIB_MEDOP_INFO {
    _ELIB_MED_ADDRESS *address;          /* Address data pointer (NULL if not required)    *2 */
    _ELIB_MED_MEMO *memo;                /* Memo data pointer (NULL if not required)        *3 */
    _ELIB_MED_IMAGE *image;             /* Image data pointer (NULL if not required)       *4 */
    _ELIB_MED_MOVIE *movie;             /* Movie data pointer (NULL if not required)       *5 */
} _ELIB_MEDOP_INFO;
```

* For each additional information storage area, **allocate the previously required size based on the calling side** and set the pointer.

Address : Allocate an area capable of storing the address data structure (_ELIB_MED_ADDRESS).

Memo : Allocate an area capable of storing the memo data structure (_ELIB_MED_MEMO).

Image : Allocate an area capable of storing the image data structure (_ELIB_MED_IMAGE).

Movie : Allocate an area capable of storing the movie data structure (_ELIB_MED_MOVIE).

(*1) If a return value indicating no registration (ELIB_DEX_NOENTRY) is returned, all of the information for which a collection request is specified has not been registered.

* Even if one information item is valid, the data is set and normal (ELIB_DEX_OK) is returned.

* If information for which collection has been requested has not been registered, NULL is set and returned for the relevant member.

(*2) Address data structure

```
typedef struct _ELIB_MED_ADDRESS {
    unsigned char    zipcode[ELIB_ZIPCODEMAX+1]; /* Zip code */
    unsigned char    data[ELIB_ADDRESSMAX+1];   /* Address data */
    unsigned short    icon;                     /* Icon information */
} _ELIB_MED_ADDRESS;
```

```
#define ELIB_ZIPCODEMAX    7    /* Size of zip code that can be registered */
```

```
#define ELIB_ADDRESSMAX    93    /* Size of address data that can be registered */
```

(*3) Memo data structure

```
typedef struct _ELIB_MED_MEMO {
    unsigned char    data[ELIB_MEMOMAX+1];      /* Memo data */
    unsigned short    icon;                     /* Icon information */
} _ELIB_MED_MEMO;
```

```
#define ELIB_MEMOMAX    100    /* Size of memo data that can be registered */
```

(*4) Image data structure

```
typedef struct _ELIB_MED_IMAGE {
    unsigned short    image_id ;                /* Image ID */
    unsigned short    icon ;                    /* Icon information */
} _ELIB_MED_IMAGE;
```


(*5) Movie data structure

```
typedef struct _ELIB_MED_MOVIE {  
    unsigned short    movie_id ;           /* Movie ID          */  
    unsigned short    icon ;              /* Icon information   */  
} _ELIB_MED_IMAGE;
```

The presence/absence of additional information registration can be identified using the additional information entry of the telephone directory data collected as described in [\[Memory Dial Read\]](#).

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate.

Note

6.14 Memory Dial Additional Information Deletion

Classification	Data exchange service support function		
Function	Memory dial additional information deletion	Symbol	Elib_DEX_Mem_DialOptionDelete
Functional overview	<p>- Deletes the additional information (address/memo/image/movie) registered in the telephone directory data of the specified memory number.</p> <p>* When deletion is completed normally, the entry value of the deleted additional information is set to OFF: No registration for the next telephone directory read operation.</p> <p>* This function is executed regardless of the previous status (Registration/No registration). Even if additional information that has not been registered is specified, it will not be handled as an error. Normal (ELIB_DEX_OK) will be returned as the return value.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Mem_DialOptionDelete (ap_id, mem_no, kind) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
mem_no	unsigned short	I	Body memory number (0 to 699) * Only the body telephone number data is valid.
kind	int	I	Specify the type of additional information to be deleted. (*1) ELIB_MEDOP_ADDR: Address ELIB_MEDOP_MEMO: Memo ELIB_MEDOP_IMAGE: Image ELIB_MEDOP_MOVIE: Movie
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark			

(*1) Additional information types can be joined using OR. For example, to specify an address and image, specify as (ELIB_MEDOP_ADDR | ELIB_MEDOP_IMAGE).

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..

Note

- When an image or movie is specified, the ID information (link information) registered in the telephone directory data is deleted. The actual image data or movie data is not deleted.

6.15 Memory Dial Registration (Main Part)

Classification	Data exchange service support function		
Function	Memory dial registration request (main part)	Symbol	Elib_DEX_Mem_DialWriteReq
Functional overview	<p>- Requests new registration or correction of the telephone directory to the specified memory number.</p> <p>- Newly registers the data if telephone directory data has not been registered to the specified memory number or updates the data if it is present.</p> <p>- The information to be written is as follows:</p> <ul style="list-style-type: none"> - Memory number - Read kana (family name) - Read kana (name) - Name (family name) - Name (name) - Telephone number count - Telephone number (4) - Mail address count - Mail address (3) - Secret information - Group number (0 to 19) - Address - Memo - Image - Movie - URL count - URL (2) 		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Mem_DialWriteReq (ap_id, mem_no, data, op_info) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
mem_no	unsigned short	I	Body memory number: 0 to 699
data	_ELIB_MEDDATA *	I	Telephone directory data structure storage address (For the data format, see [Memory Dial Read].)
op_info	_ELIB_MEDOP_INFO	I	Additional information structure (For the data format, see [Memory Dial Additional Information Collection].)
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error

Remark	
	<ul style="list-style-type: none"> - The registration counts are as follows: Telephone numbers: 700, mail addresses: 700, URL : 700, images: 100, and movies: 100. Any items exceeding these values are not registered. However, if the telephone directory has not exceeded 700 items, telephone directory data can be registered using information other than telephone numbers, mail addresses, URLs, images, and movies. - The high-level process is in charge of checking the registration count upper limit. The registration count status can be checked as described in [Memory Dial Registration Status Access]. - When registration of additional information is specified, the icon number must be specified. If the icon number is 0, processing terminates in ELIB_DEX_PRMERR (parameter error). - For the read kana, name, telephone number, and mail address, NULL is set at the end of the character string. - For the name data and read kana data, the joined family name and name are valid only up to 32 bytes. If the total of the family name and name exceeds 32 bytes, the excess amount will be truncated from the name data. - NULL is set for data that cannot be entered. <p>- Relationship with the specified limit setting</p> <p>When registering a memory dial in secret mode, cancellation of the specified limit setting will be performed within this function if a specified limit setting has been made for the telephone number to be registered.</p> <ul style="list-style-type: none"> - The temporary sort table is not updated within this function. If updating of the temporary sort table is required, call sort table update on the calling side (see [Sort Table Creation]). <p>Necessary procedure</p> <ul style="list-style-type: none"> - For each application, a MSB object must be created in advance using MsbObjectCreate. <p>Note</p> <ul style="list-style-type: none"> - When registering an image or movie, only the ID information (link information) is registered in the telephone directory data. The actual image data or movie data is not replaced and saved.

6.16 Memory Dial Registration (UIM)

Classification	Data exchange service support function		
Function	Memory dial registration (UIM)	Symbol	Elib_DEX_Uim_DialWriteReq
Functional overview	<ul style="list-style-type: none"> - Requests new registration or correction of the telephone directory to the specified memory number. - Newly registers the data if telephone directory data has not been registered to the specified memory number or corrects the data if it is present. - The information to be written is as follows: <ul style="list-style-type: none"> - Memory number - Read kana (*1) - Name (*1) - Telephone number count - Telephone number (1) (*2) - Mail address count - Mail address (1) - URL count (Invalid for UIM data) - URL (Invalid for UIM data) - Secret information (Invalid for UIM data) - Group number (20 to 30) <p>(*1) For the name and read kana, there is no distinction of the family name. The entire character string joined with the family name must be set in the family name area. In addition, all of the katakana characters of the name and read kana are stored as em-size kana characters.</p> <p>(*2) The number of valid digits of the telephone number of the UIM telephone directory data is 26. [Reference] However, the number of valid digits of the telephone number of the body (portable device) telephone directory data is 26.</p> <p>* If the telephone number exceeds 26 digits, only the first 26 digits are used. The subsequent digits are truncated.</p> <ul style="list-style-type: none"> - For the read kana, name, telephone number, and mail address, NULL is set at the end of the character string. - The valid size of the UIM data is as follows: Name: 21 bytes, read kana: 25 bytes. - NULL is set for data that cannot be entered. - The UIM read kana data is registered using em-size kana characters. - The temporary sort table is not updated within this function. If updating of the temporary sort table is required, call sort table update on the calling side (see [Sort Table Creation]). 		
Include file	srv_dex.h		

Calling sequence	int Elib_DEX_Uim_DialWriteReq (ap_id, mem_no, data) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
mem_no	unsigned short	I	UIM memory number: 700 to 749
data	_ELIB_MEDDATA *	I	Telephone directory data structure storage address (For the data format, see [Memory Dial Read] .)
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error If an error occurs or a parameter is incorrect, event notification will not be performed.

Remark

- This interface is used to request registration of one memory dial item. The result is posted by the event below after the request.

Table 3.C-5.1 Memory dial registration request result notification event format
(_ELIB_DEX_EVENT)

Member name	Type	Description
category	int	DataExchangeNotify
subtype	int	DataExcNotify_DialWriteRes (Posts the result of the memory dial registration request.)
info	int	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Failure ELIB_DEX_PBEXT_FULL: Extended telephone numbers cannot be registered
subinfo	int	This information indicates the data storage status (complete/incomplete). (*3) ELIB_DEX_COMP: Complete (All information is stored.) ELIB_DEX_INCOMP: Incomplete. Data has been dropped.
data	int	Unused (always NULL)

*3: For UIM storage data, en-size kana characters are converted to em-size kana characters and registered. At this time, however, data can be dropped.

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

6.17 Memory Dial(Main Part)

Classification	Data exchange service support function		
Function	Memory dial deletion (main part)	Symbol	Elib_DEX_Mem_DialDelReq
Functional overview	<ul style="list-style-type: none"> - Deletes the telephone directory data of the specified memory number. - When this interface is used to delete telephone directory data, the additional information is also deleted. - Even if a memory number for which data has not been registered is specified, normal (ELIB_DEX_OK) is returned as the return value. 		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Mem_DialDelReq (ap_id, mem_no) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
mem_no	unsigned short	I	Body memory number: 0 to 699
Return value	Type	I/O	Description
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark	<ul style="list-style-type: none"> - Relationship with the specified limit setting (when deleting memory dials for which a specified limit has been set) The specified limit setting is released within this function. - The temporary sort table is not updated within this function. If updating of the temporary sort table is required, call sort table update on the calling side (see [Sort Table Creation]). <p>Necessary procedure</p> <ul style="list-style-type: none"> - For each application, create a MSB object in advance using MsbObjectCreate.. <p>Note</p> <ul style="list-style-type: none"> - When a telephone directory in which an image or movie has been registered is specified, only the ID information (link information) registered in the telephone directory data is deleted. The actual image data or movie data is not deleted. 		

6.18 Memory Dial Deletion (UIM)

Classification	Data exchange service support function																				
Function	Memory dial deletion (UIM)	Symbol	Elib_DEX_Uim_DialDelReq																		
Functional overview	<div>- Deletes the telephone directory data of the specified memory number.</div> <div>- The temporary sort table is not updated within this function. If updating of the temporary sort table is required, call sort table update on the calling side (see [Sort Table Creation]).</div>																				
Include file	srv_dex.h																				
Calling sequence	int Elib_DEX_Uim_DialDelReq (ap_id, mem_no) ;																				
Argument	Type	I/O	Description																		
ap_id	unsigned int	I	Application ID																		
mem_no	unsigned short	I	UIM memory number: 700 to 749																		
Return value	Type	I/O	Description																		
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error If an error occurs or a parameter is incorrect, event notification will not be performed.																		
Remark	<div>- This interface is used to request deletion of one memory dial item. The result is posted by the event below after the request.</div> <div>Table Memory dial deletion request result notification event format (_ELIB_DEX_EVENT)</div> <table><tr><th>Member name</th><th>Type</th><th>Description</th></tr><tr><td>category</td><td>int</td><td>DataExchangeNotify</td></tr><tr><td>subtype</td><td>int</td><td>DataExcNotify_DialDelRes (Posts the result of the memory dial deletion request.)</td></tr><tr><td>Info</td><td>int</td><td>Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Failure ELIB_DEX_NOTSET: Not set (When deleting data that has not been registered)</td></tr><tr><td>subinfo</td><td>int</td><td>Unused (always NULL)</td></tr><tr><td>Data</td><td>int</td><td>Unused (always NULL)</td></tr></table>			Member name	Type	Description	category	int	DataExchangeNotify	subtype	int	DataExcNotify_DialDelRes (Posts the result of the memory dial deletion request.)	Info	int	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Failure ELIB_DEX_NOTSET: Not set (When deleting data that has not been registered)	subinfo	int	Unused (always NULL)	Data	int	Unused (always NULL)
Member name	Type	Description																			
category	int	DataExchangeNotify																			
subtype	int	DataExcNotify_DialDelRes (Posts the result of the memory dial deletion request.)																			
Info	int	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Failure ELIB_DEX_NOTSET: Not set (When deleting data that has not been registered)																			
subinfo	int	Unused (always NULL)																			
Data	int	Unused (always NULL)																			

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

6.19 Memory Dial Registration Status Request

Classification	Data exchange service support function		
Function	Memory dial registration status request	Symbol	Elib_DEX_DialCntReq
Functional overview	<p>- Returns the count of registered telephone directory.</p> <p>- The registration status of the following data can be collected.</p> <p>Telephone directory data count (MAX body: 700, UIM: 50)</p> <p>Secret count (MAX body: 700, UIM: Not a target)</p> <p>Telephone number data count (MAX body: 700, UIM: 50)</p> <p>Mail address count (MAX body: 700, UIM: 50)</p> <p>URL data count (MAX body: 700, UIM: Not a target)</p> <p>Image data count (MAX body: 100, UIM: Not a target)</p> <p>Movie data count (MAX body: 100, UIM: Not a target)</p> <p>* The telephone directory registration count indicates the total registration count including secrets.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_DialCntReq (ap_id, stg_sts, cnt) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
stg_sts	int	I	Search destination specification ELIB_DEX_MEMORY: Body ELIB_DEX_USIM: UIM
cnt	_ELIB_MEDCNT *	O	Telephone directory registration count storage address (See the Functional description.)
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMEERR: Parameter error
Remark	<p>[Telephone directory registration count information structure]</p> <pre>typedef struct _ELIB_MEDCNT { unsigned short phone; /* Telephone directory registration count */ unsigned short secret; /* Secret count (Value valid only for the body. Always 0 when UIM is specified.) unsigned short dial; /* Telephone number coun unsigned short mail; /* Mail address count unsigned</pre>		

```

short      sip;          /*Unused                               */
  unsigned short url; /* URL address count (Value valid only for the body. Always 0 when UIM is
specified.) */
  unsigned short image; /* Image data count
(Value valid only for the body. Always 0 when UIM is specified.) */
  unsigned short movie; /* Movie data count (Value valid only for the body. Always 0 when UIM is
specified.) */
} _ELIB_MEDCNT;

```

Necessary procedure

- For each application, a MSB object must be created in advance using MsbObjectCreate.

Note

6.20 Sort Table Creation

Classification	Data exchange service support function		
Function	Sort table creation	Symbol	Elib_DEX_CreateSortTableReq
Functional overview	<p>- Creates a temporary sort table based on the specified conditions and sets the current location. (Complete search, read kana data search, name search, telephone number search, group search, line search, mail address search, URL search, memory number ascending order search, and name/read kana complex search)</p> <p>- This function only creates a temporary sort table and sets the current location. To move the current location or read the memory number, use the sort table read function (see [Sort Table Read]).</p> <p>- After execution of this function, the current location is set as the beginning.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_CreateSortTableReq (ap_id, stg_sts, kind, filter, mode, sdata);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
stg_sts	Int	I	Search destination specification ELIB_DEX_MEDMIX : Body and UIM ELIB_DEX_MEMORY : Body ELIB_DEX_USIM : UIM
kind	unsigned short	I	Search type ELIB_SEARCH_ALL : Complete search ELIB_SEARCH_SOUND1 : Read kana data search (*1) ELIB_SEARCH_NAME1 : Name search (*1) ELIB_SEARCH_DIAL : Telephone number search ELIB_SEARCH_MAIL : Mail address search ELIB_SEARCH_URL : URL search ELIB_SEARCH_GROUP : Group search ELIB_SEARCH_LINE : Line search ELIB_SEARCH_MEMNUM : Memory number ascending order complete search (body only) ELIB_SEARCH_COMPLEX: Name/read kana complex search *1: The name and read kana are processed as family name joined data.
filter	unsigned short	I	Secret status, specified issuing limit status (body only *2) ELIB_ANY_SECRET : Target regardless of the secret status

			ELIB_BE_SECRET : Target if the data is secret ELIB_NO_SECRET : Target if the data is not secret ELIB_BE_LIM_OUT : Target if specified issuing limit is set for the data ELIB_NO_LIM_OUT : Target if specified issuing limit is not set for the data
mode	unsigned short	I	Search format ELIB_KEY_INCLUDE : Include search ELIB_KEY_AGREE : Previous match search (Without expanding 184,186,*31#,#31#) ELIB_KEY_AGREE_EX : Previous match search Body only (Telephone number search only) (With expanding 184,186,*31#,#31#) ELIB_KEY_QUITE : Complete match search (Telephone number search, mail address search, and URL search only) (Check until the end matches.) (Without expanding 184,186,*31#,#31#) ELIB_KEY_QUITE_EX : Complete match search Body only (Telephone number search only) (Check until the end matches.) (With expanding 184,186,*31#,#31#)
sdata	unsigned char *	I	Search data Complete search: Unused Read kana data search (family name + name): Read kana Name search (family name + name): Name Telephone number search: Telephone number Mail address search: Mail address URL search: URL Group search: Group number Line search: Line number Memory number ascending order complete

			search: Unused Name/read kana complex search: Name + read kana
Return value	Type	I/O	Description
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark			

***2: Secret status and specified issuing limit status**

The secret status and specified issuing limit status of the search type data are used to determine whether to register the data to the sort table. In addition, the specifications of the secret status and specified issuing limit status can be combined and used.

Example: Registering data set as secret but not with a specified issuing limit

ELIB_BE_SECRET | ELIB_NO_LIM_OUT

* Note: When ELIB_BE_SECRET and ELIB_BE_LIM_OUT are specified, the telephone directory in the UIM is not searched.

Table Search data detailed combinations

Search format	Include search			Previous match search (without expanding)			Previous match search (without expanding)			Complete match search (without expanding)			Complete match search (without expanding)		
	Body	UIM	Body and UIM	Body	UIM	Body and UIM	Body	UIM	Body and UIM	Body	UIM	Body and UIM	Body	UIM	Body and UIM
Search destination															
Search type															
Complete search	O	O	O	O	O	O	X	X	X	X	X	X	X	X	X
Read kana data search	O	O	O	O	O	O	X	X	X	X	X	X	X	X	X
Name search	O	O	O	O	O	O	X	X	X	X	X	X	X	X	X
Telephone number search	O	O	O	O	O	O	O	X	O	O	O	O	O	X	O
Mail address search	O	O	O	O	O	O	X	X	X	O	O	O	X	X	X
URL search	O	O	O	O	O	O	X	X	X	O	O	O	X	X	X
Group search	O	O	O	O	O	O	X	X	X	X	X	X	X	X	X
Line search	O	O	O	O	O	O	X	X	X	X	X	X	X	X	X
Memory number ascending order complete search	O	X	O	O	X	O	X	X	X	X	X	X	X	X	X
Name/read kana complex search	O	O	O	O	O	O	X	X	X	X	X	X	X	X	X

* O: Can be retrieved, X: Cannot be retrieved

* A parameter error (ELIB_DEX_PRMERR) will occur if a combination that includes an item that

cannot be retrieved is used.

* A previous match search using complete search, group search, line search, or memory number search is the same as an include search.

* If only NULL is set for the search data for other than a memory number ascending order search, the search is the same as a complete search.

Search type	Body data	UIM data
Complete search	Unused	
Read kana data search (family name + name)	Read kana specification (32Byte+NULL)	Read kana specification (25Byte+NULL)
Name search (family name + name)	Name specification (32Byte+NULL)	Name specification (21Byte+NULL)
Telephone number search	Telephone number specification (26Byte+NULL)	
Mail address search	Mail address specification (50Byte+NULL)	
URL search	URL specification (256Byte+NULL)	Unused
Group search	Group number 0 to 19 specification (1Byte)	Group number 20 to 30 specification (1Byte)
Line search	Line number 1 to 11 specification (1Byte)	
Memory number ascending order complete search	Unused	
Name/read kana complex search	Name + read kana specification (32Byte+NULL + 32Byte+NULL)	Name + read kana specification (25Byte+NULL + 21Byte+NULL)

Table Search data details

- For read kana, name, telephone number, mail address and URL, NULL is set at the end of the search character string.
- For a complete match search (end match check), only telephone number search, mail address search, and URL search apply. For a telephone number search, the end is assumed to match even if the comparison target data corresponding to the end position of the specified keyword is 'p'.
- For a name + read kana search, the character string set for sdata is the character string that joins the name and read kana, which enables previous/next match searches of the name and read kana. In addition, the character string that includes the terminating character is used to join the name and read kana.

Table Sort order (Picture writing is not a target because it cannot be entered for read kana (family name + name).)

Sequence	Contents	Details
1	Space	
2	Kana syllabary	

3	Alphabet	
4	Number	
5	Symbol	ASCII code order

Table Functions that use the temporary sort table

Function	Functional description
Sort Table Read	Moves the current location of the temporary sort table and reads one data item of the corresponding telephone directory data and the memory number.
Search Object Count Decision	Returns the number of data items registered in the temporary sort table.
Sort Table Single Deletion	Deletes the specified deletion data from the current data in the temporary sort table.
Sort Table Search	Finds which telephone directory data in the temporary sort table the telephone directory data of the specified memory number corresponds to and returns the sequence number.

Necessary procedure

- For each application, a MSB object must be created in advance using MsbObjectCreate.

Note

- If task switching causes a different task to be used for the search, the temporary sort table will be changed. When the task is switched back to the original task and the result returned as the return value, the correct telephone directory information will not be able to be read. In this case, the calling side must use the original search conditions to reconfigure the temporary sort table. This case is a restriction of the data exchange service.

6.21 Sort Table Read

Classification	Data exchange service support function		
Function	Sort table read	Symbol	Elib_DEX_ReadSortTableReq
Functional overview	<p>- Moves the current location of the temporary sort table and reads one data item of the corresponding telephone directory data and the memory number. (Complete search, read kana data search (family name/name), name search (family name/name), dial search, group search, line search, mail address search, URL search, mail number (memory number ascending order complete search), and name/read kana simultaneous search)</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_ReadSortTableReq (ap_id, stg_sts, origin, offset, data) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
stg_sts	int	I	Search destination specification ELIB_DEX_MEDMIX: Body and UIM ELIB_DEX_MEMORY: Body ELIB_DEX_USIM: UIM
origin	unsigned short	I	Reference value ELIB_ORG_SET (First data) ELIB_ORG_CUR (Current data) ELIB_ORG_END (Final data)
offset	short	I	Search offset -n: Candidate n number before n: Candidate n number later
data	_ELIB_MEDDATA *	O	Telephone directory data structure storage address (For the data format, see [Memory Dial Read] .)
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_NOTSET: Not set ELIB_DEX_PRMERR: Parameter error
Remark	<p>- This function only moves the current location and reads the memory dial of the current location. To read the data of an arbitrary memory dial, use the procedure described in [Memory Dial Read]. - The current information is updated only if this processing operation terminates normally. If the</p>		

processing operation terminates abnormally, the current information is not updated.

- To collect the additional information added to the read data, use the procedure described in [\[Memory Dial Additional Information Collection\]](#).

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- To use this function, a temporary sort table must be created and the current location set as described in sort table creation ([\[Sort Table Creation\]](#)).

Note

- If task switching causes a different task to be used for the search, the temporary sort table will be changed. When the task is switched back to the original task and the result returned as the return value, the correct telephone directory information will not be able to be read. In this case, the calling side must use the original search conditions to reconfigure the temporary sort table. This case is a restriction of the data exchange service.
- The specification of the search destination of this interface does not depend on the stg_sts parameter value. The interface operates based on the stg_sts parameter of [\[Sort Table Creation\]](#).

6.22 Memory Dial Keyword Search

Classification	Data exchange service support function		
Function	Memory dial keyword search	Symbol	Elib_DEX_Search_KeyWordReq
Functional overview	<p>- Searches using the keyword and returns the memory number of the relevant data. The search is continued up to the end of the file.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Search_KeyWordReq (ap_id, stg_sts, data) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
stg_sts	Int	I	<p>Search destination specification</p> <p>ELIB_DEX_MEMORY: Body</p> <p>ELIB_DEX_USIM: UIM</p> <p>ELIB_DEX_MEDMIX: Body->UIM</p> <p>* ELIB_DEX_MEDMIX searches for the telephone directory on the move device side. If the telephone directory is not found, it searches for the telephone directory on the UIM side.</p>
data	_ELIB_KEYWDDATA *	I	Keyword search information storage address (See the Functional description.)
Return value	Type	I/O	Description
ret	Int	O	<p>Processing result</p> <p>Memory number (≥0): Normal</p> <p>ELIB_DEX_NG: Abnormal</p> <p>ELIB_DEX_NOTSET: Not set</p> <p>ELIB_DEX_PRMERR: Parameter error</p>
Remark			

[Keyword search information structure] *1

```
typedef struct _ELIB_KEYWDDATA {
    unsigned short type;           /* Data type */
    unsigned short filter;        /* Secret status, specified issuing limit status (only body data valid *2) */
    unsigned short mode;          /* Search format */
    unsigned short origin;        /* Reference value */
    short offset;                 /* Offset */
    unsigned short direct;        /* Search direction */
    unsigned char key;            /* Keyword storage area address */
} _ELIB_KEYWDDATA;
```

(*1)

type . . ELIB_SEARCH_DIAL (Telephone number)

ELIB_SEARCH_MAIL (Mail address)

ELIB_SEARCH_URL (URL address)

* When ELIB_SEARCH_URL (URL address) is specified, a parameter error (ELIB_DEX_PRMERR) will be returned if a value other than body (ELIB_DEX_MEMORY) is specified for the search destination (stg_sts).

filter . . ELIB_ANY_SECRET: Target regardless of the secret status

ELIB_BE_SECRET: Target if the data is secret

ELIB_NO_SECRET: Target if the data is not secret

ELIB_BE_LIM_OUT: Target if specified issuing limit is set for the data

ELIB_NO_LIM_OUT: Target if specified issuing limit is not set for the data

mode . . ELIB_KEY_QUITE (Keyword complete match without expanding 184/186/*31#/#31#)

ELIB_KEY_QUITE_EX (Keyword complete match with expanding 184/186/*31#/#31#)

Body data only

origin . . ELIB_ORG_SET (First data)

ELIB_ORG_END (Final data)

offset . . -n: Search from n number before the reference value

0: Search from the reference value

n: Search from n number after the reference value

direct . . ELIB_TO_TOP (Forward direction)

ELIB_TO_BTM (Backward direction)

key . . Keyword storage area address

*2: Secret status and specified issuing limit status

The secret status and specified issuing limit status of the search type data are used to determine whether to perform the search. In addition, the specification of the secret status and specified issuing limit status can be combined and used.

Example: Registering data set as secret but not with a specified issuing limit

ELIB_BE_SECRET | ELIB_NO_LIM_OUT

* Note: When ELIB_BE_SECRET and ELIB_BE_LIM_OUT are specified for the search, the UIM data will not be a target of the search.

- NULL is set at the end of the keyword character string.

- For keyword complete matching, matching is assumed only if the end of the keyword and end of the comparison target data match. However, for a telephone number search, the end is assumed to match even if the comparison target data corresponding to the end position of the keyword is 'p'.

- This keyword search is used for searching for data not in the sort table but in real

memory. As a result, there is no influence on the current location set as described in [\[Sort Table Creation\]](#) or [\[Sort Table Read\]](#).

- This function first searches for the offset location. The function then searches forward of backward from the offset location and returns the memory number if it finds the relevant data.
- This function collects only the memory numbers of specified locations. To read memory dial data, use the procedure described in [\[Memory Dial Read\]](#).

Necessary procedure

- For each application, a MSB object must be created in advance using MsbObjectCreate.

Note

6.23 Memory Number Check

Classification	Data exchange service support function		
Function	Memory number check	Symbol	Elib_DEX_MemNum_ChkReq
Functional overview	<p>- Checks the usage status of the memory number.</p> <p>- Checks the status of the attribute information (auto display and secret setting) of the specified memory number.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_MemNum_ChkReq (ap_id, mem_no, flag) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
mem_no	unsigned short	I	Memory number (0 to 749) Body memory number: 0 to 699 UIM memory number: 700 to 749
Flag	unsigned short *	O	Memory dial attribute information storage address (body only) * The value is undefined when UIM is specified.
Return value	Type	I/O	Description
Ret	int	O	Processing result ELIB_DEX_USED: Being used ELIB_DEX_UNUSED: Unused ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark			

- If the specified memory number is being used, used (ELIB_DEX_USED) is returned as the return value.
- If the specified memory number is not being used, unused (ELIB_DEX_UNUSED) is returned as the return value.
- The auto display and secret setting states are stored in the memory dial attribute information (flag) as bit information.
- The bit positions are given in the table below. For the secret setting, bit on means that secret is set and bit off means that secret is not set.

Table Bit position information

bit	Definition	Description
bit0 to 3	-	: Reserve
bit4 to 6	ELIB_MC_FBIT_AUTOX(x)	: Auto display setting (0: Not set, 1 to 4: Telephone number multi ID) * The macro to the left is used to collect the multi-number.
bit7	ELIB_FBIT_SECRET	: Secret setting
bit8	-	: Reserve
bit9	-	: Reserve
bit10	-	: Reserve
bit11	-	: Reserve
bit12	-	: Reserve
bit13	-	: Reserve
bit14	-	: Reserve
bit15	-	: Reserve

- * The auto display and secret setting functions apply only for telephone directory data on the body side. Therefore, if this interface is used to specify a UIM memory number, the memory dial information storage address in the return value will be an undefined value.

Necessary procedure

- For each application, a MSB object must be created in advance using MsbObjectCreate.

Note

6.24 Memory Number Request

Classification	Data exchange service support function		
Function	Memory number request	Symbol	Elib_DEX_MemNum_GetReq
Functional overview	<p>- Searches for and collects the newest and last numbers of the unused memory numbers.</p> <p>*1: The body data and UIM data are searched for separately (mixed searching using body and UIM data is not supported).</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_MemNum_GetReq (ap_id, stg_sts, kind, start_no, end_no) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
stg_sts	int	I	Search destination specification ELIB_DEX_MEMORY: Body ELIB_DEX_USIM: UIM
kind	unsigned short	I	Collection memory number type ELIB_INDEX_LEAST (Newest number search) ELIB_INDEX_LAST (Last number search)
Start_no	unsigned short	I	Search range start index number ELIB_DEX_MEMORY valid range: 0 to 699 (*1) ELIB_DEX_USIM valid range: 700 to 749
end_no	unsigned short	I	Search range end index number ELIB_DEX_MEMORY valid range: 0 to 699 (*1) ELIB_DEX_USIM valid range: 700 to 749
Return value	Type	I/O	Description
ret	int	O	Processing result Memory number (≥0): Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMEERR: Parameter error
Remark			

- Search range start (start_no) and end (end_no) are also search targets.
- If the search result is no good, abnormal end (ELIB_DEX_NG) is returned.
- If the search result is good, the retrieved newest and last numbers are set in the return value.
- This processing operation does not move the current location.

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..

Note

6.25 Memory Dial Attribute Setting/Release

Classification	Data exchange service support function		
Function	Memory dial attribute setting/release	Symbol	Elib_DEX_MemAttr_SetReq
Functional overview	<p>- Sets/releases the attribute information (auto display and secret) of the memory dial.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_MemAttr_SetReq (ap_id, mem_no, kind, mode, multino) ;		
Argument	Type	I/O	Description
Ap_id	unsigned int	I	Application ID
Mem_no	unsigned short	I	Body memory number: 0 to 699
kind	unsigned short	I	Attribute type ELIB_ATR_AUTO: Auto display ELIB_ATR_SECRET: Secret setting
mode	unsigned short	I	Set/release ELIB_FIL_SET: Set ELIB_FIL_CANCEL: Release
multino	unsigned char	I	Multi-number (valid range: 0 to 3) When setting/releasing auto display, specify the multi-number in the telephone directory of the target telephone number.
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_NOTSET: Not set ELIB_DEX_PRMERR: Parameter error
Remark	<p>- Auto display is set for telephone numbers. Secret is set for memory dials.</p> <p>- When setting/releasing auto display, abnormal end (ELIB_DEX_NG) will be returned as the return value and set/release will not be executed if the target telephone number is not present in the specified memory dial.</p> <p>- Relationship with the specified limit setting (when secret has been set for memory dials for which a specified limit has been set) The specified limit setting is released within this function.</p>		

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..

Note

6.26 Search Object Count Request

Classification	Data exchange service support function		
Function	Search object count Request	Symbol	Elib_DEX_CountObjectReq
Functional overview	<p>- Returns the number of data items registered in the temporary sort table.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_CountObjectReq (ap_id, stg_sts, kind) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
stg_sts	int	I	Telephone directory specification ELIB_DEX_MEDMIX: Body and UIM ELIB_DEX_MEMORY: Body ELIB_DEX_USIM: UIM
kind	unsigned short	I	Telephone directory count collection type (Valid only when body is specified.) ELIB_SEARCH_PHONE: Telephone directory data
Return value	Type	I/O	Description
ret	int	O	Processing result Registration count (≥ 0): Normal ELIB_DEX_NG: Abnormal ELIB_DEX_NOTSET: Not set ELIB_DEX_PRMERR: Parameter error
Remark	<p>- If current information of the temporary sort table has not been set, not set (ELIB_DEX_NOTSET) will be returned as the return value.</p> <p>- The telephone number count and mail address count are not registered to the temporary sort table.</p> <p>- The additional information (address, memo, and image) is not registered to the temporary sort table because there is no search function for them.</p> <p>Necessary procedure</p> <p>- For each application, create a MSB object in advance using MsbObjectCreate..</p> <p>- To use this function, a temporary sort table must be created and the current location set as described in sort table creation ([Sort Table Creation]).</p> <p>Note</p> <p>- If task switching causes a different task to be used for the search, the temporary sort table will be changed. When the task is switched back to the original task and the result returned as the return</p>		

value, the correct telephone directory information will not be able to be read. In this case, the calling side must use the original search conditions to reconfigure the temporary sort table. This case is a restriction of the data exchange service.

- The specification of the search destination of this interface does not depend on the stg_sts parameter value. The interface operates based on the stg_sts parameter of [\[Sort Table Creation\]](#).

6.27 Sort Table Single Data Item Deletion

Classification	Data exchange service support function		
Function	Sort table single data item deletion	Symbol	Elib_DEX_DeISortTableReq
Functional overview	<p>Deletes the specified deletion data from the current data of the temporary sort table.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_DeISortTableReq (ap_id, stg_sts, kind, multino) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
stg_sts	int	I	Telephone directory specification ELIB_DEX_MEDMIX: Body and UIM ELIB_DEX_MEMORY: Body ELIB_DEX_USIM: UIM
kind	unsigned short	I	Deletion data specification ELIB_DEL_PHONE: Telephone directory data ELIB_DEL_DIAL: Telephone number data ELIB_DEL_MAIL: Mail address data ELIB_DEL_URL_IMODE: URL data
multino	unsigned short	I	Deletion multi-number (Valid only for body data) Valid range Telephone number: 0 to 3 Mail: 0 to 2 URL: 0 and 1
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_NOENTRY: No entry ELIB_DEX_PRMERR: Parameter error
Remark			

- There are four types of deletion data: Telephone directory data, telephone number data, mail address data, and URL data.
- If the deletion data information is telephone directory data, the deletion multi-number setting will be ignored.
- If the deletion data information is telephone number data, 0 to 3 are valid for the deletion multi-number setting.
- If the deletion data information is mail address data, 0 to 2 are valid for the deletion multi-number setting.
- If the deletion data information is URL data, 0 and 1 are valid for the deletion multi-number setting.
- If the deletion multi-number setting value is not within the valid range or there is no data at the specified location, no entry (ELIB_DEX_NOENTRY) will be set.

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- To use this function, a temporary sort table must be created and the current location set as described in sort table creation ([\[Sort Table Creation\]](#)).

Note

- If task switching causes a different task to be used for the search, the temporary sort table will be changed. When the task is switched back to the original task and the result returned as the return value, the correct telephone directory information will not be able to be read. In this case, the calling side must use the original search conditions to reconfigure the temporary sort table. This case is a restriction of the data exchange service.
- The specification of the search destination of this interface does not depend on the stg_sts parameter value. The interface operates based on the stg_sts parameter of [\[Sort Table Creation\]](#).

6.28 Sort Table Search

Classification	Data exchange service support function		
Function	Sort table search	Symbol	Elib_DEX_SearchSortTable
Functional overview	<p>- Finds which telephone directory data in the sort table corresponds to the telephone directory data of the specified memory number and returns the sequence number.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_SearchSortTable (ap_id, mem_no) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
mem_no	unsigned short	I	Memory number (0 to 749) Body memory number: 0 to 699 UIM memory number: 700 to 749
Return value	Type	I/O	Description
ret	Int	O	Processing result Sequence number (≥0): Normal ELIB_DEX_NG: Abnormal ELIB_DEX_NOENTRY: No entry ELIB_DEX_PRMERR: Parameter error
Remark	<p>- This function searches for the relevant memory number from the beginning up to the end of the temporary sort table. If the result of the search is no registration count or the relevant data cannot be found, no entry (ELIB_DEX_NOENTRY) will be returned.</p> <p>Necessary procedure</p> <ul style="list-style-type: none"> - For each application, create a MSB object in advance using MsbObjectCreate.. - To use this function, a temporary sort table must be created and the current location set as described in sort table creation ([Sort Table Creation]). <p>Note</p> <ul style="list-style-type: none"> - If task switching causes a different task to be used for the search, the temporary sort table will be changed. When the task is switched back to the original task and the result returned as the return value, the correct telephone directory information will not be able to be read. In this case, the calling side must use the original search conditions to reconfigure the temporary sort table. This case is a restriction of the data exchange service. 		

6.29 Priority Telephone Number Search

Classification	Data exchange service support function		
Function	Priority telephone number search	Symbol	Elib_DEX_Search_PreferenceDial
Functional overview	<p>- Finds the telephone numbers registered in the telephone directory that completely match the specified keyword without expanding the additional numbers and returns the memory number and multi-number (0 to 3) of the relevant telephone numbers.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Search_PreferenceDial (ap_id, data) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
data	_ELIB_PRISEARCH *	I/O	Priority search structure address
Return value	Type	I/O	Description
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_NOENTRY: No entry ELIB_DEX_PRMERR: Parameter error
Remark	<pre> [Priority search information structure][I/O] typedef struct _ELIB_PRISEARCH { int stg_sts; /* Telephone directory specification */ unsigned short filter; /* Secret status and specified issuing limit status (valid only for body) */ unsigned char key; /* Keyword storage area */ unsigned short memno; /* Relevant memory number */ unsigned short multi; /* Relevant multi-number */ } _ELIB_PRISEARCH; Input information [I/-] stg_sts . . ELIB_DEX_MEMORY: Body only search ELIB_DEX_USIM: UIM only search ELIB_DEX_MIXA: Search in order of body -> UIM ELIB_DEX_MIXB: Search in order of UIM -> body filter . . ELIB_ANY_SECRET: Target regardless of the secret status EIIB_BE_SECRET: arget if the data is secret </pre>		

	ELIB_NO_SECRET:	Target if the data is not secret
	ELIB_BE_LIM_OUT:	Target if specified issuing limit is set for the data
	ELIB_NO_LIM_OUT:	Target if specified issuing limit is not set for the data
Key	. . Keyword storage area address	

Output information [-/O]

memno i|i|Body memory number: 0 to 699
 UIM memory number: 700 to 749
 multi . . Telephone number: 0 to 3 (UIM specified: 0)
 Mail address: 0 to 2 (UIM specified: 0)

- NULL is set at the end of the keyword character string.
- This function performs two branch searches from the telephone number sort table. The function collects the memory number and multi-number (0 to 3) of the relevant telephone number for which the secret setting status and secret information match.
- If there is no telephone number registration count, the specified limit setting status conflicts with the telephone number specified limit setting information, or the relevant telephone number cannot be found, no entry (ELIB_DEX_NOENTRY) will be returned. If the secret setting and specified limit setting are incorrect, parameter error (ELIB_DEX_PRMERR) will be returned.

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate.

Note

6.30 Priority Mail Address Search

Classification	Data exchange service support function		
Function	Priority mail address search	Symbol	Elib_DEX_Search_PreferenceMail
Functional overview	<p>- Finds the mail addresses registered in the telephone directory that completely match the specified keyword and returns the memory number and multi-number (0 to 2) of the relevant mail addresses.</p> <p>- If this function cannot find the relevant data in the first round, it searches using the keyword corrected using the data type of the second round mail address.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Search_PreferenceMail (ap_id, data) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
data	_ELIB_PRISEARCH *	I/O	Priority search structure address (For the data format, see [C-18. Priority Telephone Number Search].)
Return value	Type	I/O	Description
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_NOENTRY: No entry ELIB_DEX_PRMERR: Parameter error
Remark	<p>- NULL is set at the end of the keyword character string.</p> <p>- If there is no mail address registration count or the relevant mail address cannot be found, no entry (ELIB_DEX_NOENTRY) will be returned. If the secret setting is incorrect, parameter error (ELIB_DEX_PRMERR) will be returned.</p> <p>Necessary procedure</p> <p>- For each application, a MSB object must be created in advance using MsbObjectCreate.</p> <p>Note</p>		

6.31 Set/Release Specified Function Request

Classification	Data exchange service support function		
Function	Set/release specified function request	Symbol	Elib_DEX_Limit_Set
Functional overview	<p>- Sets/releases the specified limit setting/specified voice monitor function of the specified memory dial.</p> <p>The UIM telephone directory data is not a target.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Limit_Set(ap_id, mem_no, multino, kind, value, mode);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
mem_no	unsigned short	I	Mail number/group number (*1) Mail number: 0 to 699 Group number: 1 to 19
multino	unsigned char	I	Telephone number/mail address/multi-number (*2) Telephone number: 0 to 3 Mail address; 0 to 2
kind	unsigned short	I	Specified function type
value	unsigned short	I	Specified value number (*3) When the specified function type is specified handy function, enable 0 (off) or specify the number.
mode	unsigned short	I	Set/release ELIB_FIL_SET: Set ELIB_FIL_CANCEL: Release
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error ELIB_DEX_NOTSET: Not set ELIB_DEX_FULL: Limit exceeded ELIB_DEX_DUPLICATE: Duplicate
Remark			

*1: If the specified function type kind is not a group specified handy function, the number is handled as a memory number. If kind is a group specified handy function, the number is handled as a group number.

*2: The telephone number (0 to 3)/mail address (0 to 2) of the multi-number specified here are targets of set/release.

- Mail specified ring tone and mail specified illumination -> The mail address multi-number is stored.

- Other than above -> The telephone number multi-number is stored.

* If the specified function type kind is a group specified handy function, no evaluation is performed.

*3: For the specified value number (value), 0 is handled as function OFF. Specify each setting value as follows:

- For the specified ring tone setting, specify a sound ID provided by the sound service. A specified ring tone can also be specified for the javaapplication sound. Specify a sound ID provided by the ELIB sound service as the function number. Specify a movie ID provided by the movie control service. The sound service and movie control service manage the sound ID and movie ID so that they do not conflict.
- For the specified incoming image setting, specify an image display data retention number provided by the resource management library. User-defined animation numbers provided by the resource management library are also allocated for user-defined animations set in the specified incoming images. Therefore, specify that value as the function number.
- For the specified illumination setting, specify a LED color specified value provided by the LMP management service.
- For the specified response message setting, specify a data number provided by the record and playback service.
- For the videophone outgoing communication speed setting, specify ELIB_DEX_TVSPPEED_32K or ELIB_DEX_TVSPPEED_64K.

* If an attempt is made to specify a value when the maximum registration count of the specified function has been reached, limit exceeded will be returned as the return value. The value will not be set.

* If a dial attempting to be set has already been registered, duplicate will be returned as the return value. Duplicate is assumed when matching data is found as the result of executing the search operations listed in the table below.

Table 3.C-20.4 Search processing table

Search data (*6)	Specified function type specified in argument kind	Search processing
Telephone number data	Specified limit setting	The telephone number data (*4) of the retrieval data is retrieved from telephone number data (*4) that has already been set. The telephone number data here means the telephone number data processed using the data described

		in note *4 below.
	Specified handy function	The telephone number data (*4) of the retrieval data is retrieved from telephone number data (*4) that has already been set. The telephone number data here means the telephone number data processed using the data described in note *4 below.
Mail address	Specified limit setting	-
	Specified handy function	The mail address data of the retrieval data is retrieved from mail address data that has already been set.

*4: Telephone number data from which 184, 186, *31#, #31#, and characters following 'p' from the front have been deleted.

*6: For C-20: Data specified in argument mem_no
For C-23: Data specified in argument key

* This function does not perform exclusive control (specified incoming refuse/permit and specified incoming transfer and specified caretaking telephone) for the specified limit setting. Therefore, a specified function status check using an application ([\[Specified Function Status Check\]](#) ELIB_DEX_Limit_Chk) must be used to check.

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..

Note

6.32 Count Specified Function Request

Classification	Data exchange service support function		
Function	Count specified function request	Symbol	Elib_DEX_Limit_Cnt
Functional overview	<p>- Returns the specified limit setting/specified handy function count.</p> <p>* UIM telephone directory data is not a target.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Limit_Cnt(ap_id, kind , cnt);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
kind	unsigned short	I	Specified function type
cnt	unsigned short *	O	Specified function count storage address
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark	<p>Necessary procedure</p> <p>- For each application, create a MSB object in advance using MsbObjectCreate..</p> <p>Note</p>		

6.33 Enable/Disable Specified function Request

Classification	Data exchange service support function										
Function	Enable/disable specified function request	Symbol	Elib_DEX_Limit_Exist								
Functional overview											
<p>- Returns each count for all specified limit settings/specified voice monitor function types.</p> <p>* UIM telephone directory data is not a target.</p>											
Include file	srv_dex.h										
Calling sequence	int Elib_DEX_Limit_Exist(ap_id, flag, ex_flag, exist);										
Argument	Type	I/O	Description								
ap_id	unsigned int	I	Application ID								
flag	unsigned long	I	Specified function type (multiple settings allowed) (*1)								
ex_flag	unsigned long	I	Specified function type (multiple settings allowed) (*2)								
exist	unsigned short *	O	Specified function count storage address (*3)								
Return value	Type	I/O	Description								
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error								
Remark											
<p>*1: In the specified function type storage area, turn on the bit of the item to be checked (multiple settings allowed).</p> <p>*2: In the specified function type storage area, turn on the bit of the item to be checked (multiple settings allowed).</p> <p>*3: In the relevant area, prepare an array of size ELIB_DEX_LIMIT_EXISTSIZE.</p> <p>-> Because this is an unsigned short array, the size becomes (ELIB_DEX_LIMIT_EXISTSIZE x 2) bytes. For the requested specified function type, collect a count and set the relevant number of bytes for the relevant area. Regardless of the specified function set in the flag, the location where the count of each function is stored is fixed in this area. The locations are allocated as described below.</p> <table><tr><td>Specified limit setting</td><td>Specified issuing limit</td></tr><tr><td>ELIB_LIM_OUT</td><td>Specified incoming permission</td></tr><tr><td>ELIB_LIM_PERMISSION</td><td>Specified incoming refuse</td></tr><tr><td>ELIB_LIM_REFUSE</td><td></td></tr></table>				Specified limit setting	Specified issuing limit	ELIB_LIM_OUT	Specified incoming permission	ELIB_LIM_PERMISSION	Specified incoming refuse	ELIB_LIM_REFUSE	
Specified limit setting	Specified issuing limit										
ELIB_LIM_OUT	Specified incoming permission										
ELIB_LIM_PERMISSION	Specified incoming refuse										
ELIB_LIM_REFUSE											

ELIB_LIM_CARETAKING	Specified caretaking telephone
ELIB_LIM_REMOVE	Specified incoming transfer
Specified handy function	
ELIB_CNV_MELODY	Specified ring tone
ELIB_CNV_DIALMELODY	Mail incoming ring tone set for the telephone number
ELIB_CNV_MAILMELODY	Mail incoming ring tone set for the mail address
ELIB_CNV_LIGHT	Specified illumination
ELIB_CNV_DIALLIGHT	Mail specified illumination for the telephone number
ELIB_CNV_MAILLIGHT	Mail specified illumination set for the mail address
ELIB_CNV_IMAGE	Specified incoming image
ELIB_CNV_MESSAGE	Specified response message
ELIB_CNV_TVSPPEED	Videophone outgoing communication speed setting
Group specified handy function	
ELIB_CNV_GROUPELODY	Group specified ring tone
ELIB_CNV_GROUPELODY	Group mail specified ring tone
ELIB_CNV_GROUPLIGHT	Group specified illumination
ELIB_CNV_GROUPLIGHT	Group mail specified illumination
ELIB_CNV_GROUPELODY	Group specified incoming image
ELIB_CNV_GROUPELODY	Group specified response message
ELIB_CNV_GROUPELODY	Group videophone outgoing communication speed setting

Necessary procedure

- For each application, a MSB object must be created in advance using MsbObjectCreate.

Note

6.34 Specified Function Status Check

Classification	Data exchange service support function		
Function	Specified function status check	Symbol	Elib_DEX_Limit_Chk
Functional overview	<p>- Checks the status of the specified limit setting/specified voice monitor function of the specified dial.</p> <p>- For the decision on whether the data set in the argument key has been set, see C-20 Remark column of "[Table 3.C-20.4 Search processing table]." In addition, the decision that the data has been set is made when matching data is found as the result of executing the searches in the search processing table.</p> <p>* UIM telephone directory data is not a target.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Limit_Chk(ap_id, kind, key, func);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
kind	unsigned short	I	Specified function type
key	unsigned char *	I	Telephone number/mail address/group number storage area address (*1)
func	void *	O	Specified function information storage area address (*2)
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error ELIB_DEX_NOENTRY: No entry
Remark			

*1: The specified function type is used to judge and recognize whether the setting value is a telephone number, mail address, or group number.

- Mail specified ring tone or mail specified illumination -> The mail address is stored.
- Group specified handy function -> The group number is stored.
- Other than the above -> The telephone number is stored.

*2: The specified function information storage area is defined using the following data configurations:

(1) If the specified function type is classified by a specified limit setting

/* Information section structure of specified function information without a setting value */

```
typedef struct _ELIB_FUNC_INFO
{
    unsigned short    memno;        /* Memory number          */
    unsigned short    index;        /* Telephone number
multi-number          */
    unsigned short    flg;          /* Telephone number flag   */
}_ELIB_FUNC_INFO;
```

(2) If the specified function type is classified by a specified handy function

/* Information section structure of specified function information with a setting value */

```
typedef struct _ELIB_VALUE_INFO
{
    unsigned short    memno;        /* Memory number          */
    unsigned short    index;        /* Telephone number/mail address
/* Multi-number(* 2-1)*/
    unsigned short    value;        /* Setting value          (* 2-2)*/
}_ELIB_VALUE_INFO;
```

(3) If the specified function type is classified by a specified group specified handy function

/* Information section structure of specified function information with a setting value */

```
typedef struct _ELIB_VALUE_INFO
{
    unsigned short    memno;        /* Group number          */
    unsigned short    index;        /* Unused                */
    unsigned short    value;        /* Setting value          (* 2-2) */
}_ELIB_VALUE_INFO;
```

*2-1: The specified function type is used to indicate the multi-number of the telephone number or mail address.

- Mail specified ring tone or mail specified illumination -> The mail address multi-number is stored.
- Other than the above -> The telephone number multi-number is stored.

*2-2: For the setting value (value), 0 is handled as function OFF. Specify each setting value as follows:

- For the specified ring tone setting, specify a sound ID provided by the sound service. A specified ring tone can also be specified for the javaapplication sound. Specify a sound ID provided by the ELIB sound service as the function number. Specify a movie ID provided by the movie control service. The sound service and movie control service manage the sound ID and movie ID so that they do not conflict.
- For the specified incoming image setting, specify an image display data retention number provided by the resource management library. User-defined animation numbers provided by the resource management library are also allocated for user-defined animations set in the specified incoming images. Therefore, specify that value as the function number.
- For the specified illumination setting, specify a LED color specified value provided by the LMP management service.
- For the specified response message setting, specify a data number provided by the record and playback service.
- For the videophone outgoing communication speed setting, specify
ELIB_DEX_TVSPPEED_32K or ELIB_DEX_TVSPPEED_64K.

Necessary procedure

- For each application, a MSB object must be created in advance using MsbObjectCreate.

Note

6.35 Initialize Specified Function Information

Classification	Data exchange service support function		
Function	Initialize specified function information	Symbol	Elib_DEX_Limit_Init
Functional overview	<p>- Initializes the specified limit setting/specified voice monitor function information.</p> <p>* UIM telephone directory data is not a target.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Limit_Init(ap_id, kind, ex_kind);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
kind	unsigned long	I	Specified function initialization item
ex_kind	unsigned long	I	Specified function initialization item
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark	<p>Necessary procedure</p> <p>- For each application, create a MSB object in advance using MsbObjectCreate..</p> <p>Note</p>		

6.36 Clear Specified Function Condition

Classification	Data exchange service support function		
Function	Clear specified function condition	Symbol	Elib_DEX_Limit_Clr
Functional overview	<p>- Initializes the data that has the same value as the setting value of the initialization object setting value (value) of the specified functions specified using the specified function initialization item (kind).</p> <p>* UIM telephone directory data is not a target.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Limit_Clr(ap_id, kind, value);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
kind	unsigned short	I	Specified function initialization item
value	unsigned short	I	Initialization target setting value (*1) (*2)
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark			

*1: Only the data for which the same value as that of the initialization object setting value (value) is set is initialized.

*2: For the initialization object setting value (value), 0 is handled as function OFF. Specify each setting value as follows:

- For the specified ring tone setting, specify a sound ID provided by the sound service. A specified ring tone can also be specified for the javaapplication sound. Specify a sound ID provided by the ELIB sound service as the function number. Specify a movie ID provided by the movie control service. The sound service and movie control service manage the sound ID and movie ID so that they do not conflict.
- For the specified incoming image setting, specify an image display data retention number provided by the resource management library. User-defined animation numbers provided by the resource management library are also allocated for user-defined animations set in the specified incoming images. Therefore, specify that value as the function number.
- For the specified illumination setting, specify a LED color specified value provided by the LMP management service.
- For the specified response message setting, specify a data number provided by the record and playback service.
- For the videophone outgoing communication speed setting, specify ELIB_DEX_TVSPPEED_32K or ELIB_DEX_TVSPPEED_64K.

Necessary procedure

- For each application, a MSB object must be created in advance using MsbObjectCreate.

Note

6.37 Request Specified Function Setting Enabled/Disabled

Classification	Data exchange service support function		
Function	Request specified function setting enabled/disabled	Symbol	Elib_DEX_Limit_Ref
Functional overview	<p>- Checks whether the same value as that of the setting value of the type information (value) has been set in the specified functions specified using the specified function initialization item (kind).</p> <p>* UIM telephone directory data is not a target.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Limit_Ref(ap_id, kind, value);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
kind	unsigned short	I	Specified function type
value	unsigned short	I	Type information (*1) Specify the setting value to be retrieved.
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_FIND: Setting data found ELIB_DEX_NOTFIND: No setting data ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark	<p>*1: For the type information (value), 0 is handled as function OFF. Specify each setting value as follows:</p> <ul style="list-style-type: none"> - For the specified ring tone setting, specify a sound ID provided by the sound service. A specified ring tone can also be specified for the javaapplication sound. Specify a sound ID provided by the ELIB sound service as the function number. Specify a movie ID provided by the movie control service. The sound service and movie control service manage the sound ID and movie ID so that they do not conflict. - For the specified incoming image setting, specify an image display data retention number provided by the resource management library. User-defined animation numbers provided by the resource management library are also allocated for user-defined animations set in the specified incoming images. Therefore, specify that value as the 		

function number.

- For the specified illumination setting, specify a LED color specified value provided by the LMP management service.
- For the specified response message setting, specify a data number provided by the record and playback service.
- For the videophone outgoing communication speed setting, specify ELIB_DEX_TVSPPEED_32K or ELIB_DEX_TVSPPEED_64K.

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..

Note

6.38 Check Specified Incoming Image Setting

Classification	Data exchange service support function		
Function	Check specified incoming image setting	Symbol	Elib_DEX_Limit_OrgImageCheck
Functional overview	<p>- Decides whether original images or user-defined movies have been set as specified incoming images.</p> <p>- This function provides the processing necessary to determine whether a specific image has been set or not set as an incoming image.</p> <p>* UIM telephone directory data is not a target.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Limit_OrgImageCheck (ap_id, kind, ex_kind, flg);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
kind	unsigned long	I	Specified function type
ex_kind	unsigned long	I	Specified function type
flg	unsigned char *	O	Image specified setting status (*1)
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark			

*1: The setting status is set using a byte array as follows:

Setting status bitXon = Setoff = Not set

flg[0] bit0 (image 01) to bit7 (image 08)

flg[1] bit0 (image09) to bit7 (image 16)

: :

flg[23]bit0 (image 185) to bit7 (image 192)

flg[24]bit0 (image 193) to bit7 (image 200)

: :

flg[49]bit0 (image 383) to bit7 (image 400)

flg[50]bit0 (user-defined animation 01) to bit7 (user-defined animation 08)

flg[51]bit0 (user-defined animation 09) to bit7 (user-defined animation 16)

flg[52]bit0 (user-defined animation 17) to bit3 (user-defined animation 20)

* Bit 4 and subsequent bits of flg[52] are not used (always off).

* An array of sufficient size capable of outputting the results of 400 original image registrations and 20 user-defined animations must be prepared on the calling side.

Necessary procedure

- For each application, a MSB object must be created in advance using MsbObjectCreate.

Note

6.39 Change Specified Incoming Image

Classification	Data exchange service support function		
Function	Change specified incoming image	Symbol	Elib_DEX_Limit_ImageChg
Functional overview	<p>- When the setting value of the specified handy function corresponding to the specified function conversion item (kind) is a conversion object original image, this function converts to the setting value (value) after conversion.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Limit_ImageChg (ap_id, kind, ex_kind, flg, value);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
kind	unsigned long	I	Specified function conversion item
ex_kind	unsigned long	I	Specified function conversion item
flg	unsigned char *	I	Conversion target original image specified bit string (*1)
value	unsigned short	I	Setting value after conversion
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark			

*1: The conversion object original image specified bit string is set as follows:

Setting status bitXon = Setoff = Not set
flg[0] bit0 (image 01) to bit7 (image 08)
flg[1] bit0 (image 09) to bit7 (image 16)
:
:
flg[23]bit0 (image 185) to bit7 (image 192)
flg[24]bit0 (image 193) to bit7 (image 200)
:
:
flg[49]bit0 (image 383) to bit7 (image 400)
flg[50]bit0 (user-defined animation 01) to bit7 (user-defined animation 08)
flg[51]bit0 (user-defined animation 09) to bit7 (user-defined animation 16)
flg[52]bit0 (user-defined animation 17) to bit3 (user-defined animation 20)

* Bit 4 and subsequent bits of flg[52] are not used (always off).

* An array of sufficient size capable of outputting the results of 400 original image registrations and 20 user-defined animations must be prepared on the calling side.

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..

Note

6.40 Create Specified Function Sort Table

Classification	Data exchange service support function		
Function	Create specified function sort table	Symbol	Elib_DEX_Limit_CreateSort
Functional overview	<p>- Creates a temporary sort table and sets the current location based on the specified search conditions. The sort table is prepared exclusively for the specified functions separate from the telephone directory sort table.</p> <p>- This function only creates a temporary sort table and sets the current location. To move the current location or read the specified function registration number, use the sort table read function.</p> <p>- After execution of this function, the current location is set as the beginning.</p> <p>* UIM telephone directory data is not a target.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Limit_CreateSort(ap_id, kind, value);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
kind	unsigned short	I	Specified function type
value	unsigned short	I	Type information (*1) Specify the setting value to be retrieved.
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark			

*1: For the type information (value), 0 is handled as function OFF. Specify each setting value as follows:

- For the specified ring tone setting, specify a sound ID provided by the sound service. A specified ring tone can also be specified for the javaapplication sound. Specify a sound ID provided by the ELIB sound service as the function number. Specify a movie ID provided by the movie control service. The sound service and movie control service manage the sound ID and movie ID so that they do not conflict.
- For the specified incoming image setting, specify an image display data retention number provided by the resource management library. User-defined animation numbers provided by the resource management library are also allocated for user-defined animations set in the specified incoming images. Therefore, specify that value as the function number.
- For the specified illumination setting, specify a LED color specified value provided by the LMP management service.
- For the specified response message setting, specify a data number provided by the record and playback service.
- For the videophone outgoing communication speed setting, specify
ELIB_DEX_TVSPPEED_32K or ELIB_DEX_TVSPPEED_64K.

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..

Note

- If task switching causes a different task to be used for the search, the temporary sort table will be changed. When the task is switched back to the original task and the result returned as the return value, the correct specified function information will not be able to be read. In this case, the calling side must use the original search conditions to reconfigure the temporary sort table. This case is a restriction of the data exchange service.

6.41 Read Specified Function Sort Table

Classification	Data exchange service support function		
Function	Read specified function sort table	Symbol	Elib_DEX_Limit_ReadSort
Functional overview	<p>- Moves the current location for the sort table created at sort table creation and reads the specified function registration number of the current location and the telephone directory data of the registered telephone number/mail address set for the specified function. The sort table is prepared exclusively for the specified functions separate from the telephone directory sort table.</p> <p>- This function moves the current location only if processing terminates normally. The function does not move the current location if processing terminates abnormally.</p> <p>* UIM telephone directory data is not a target.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Limit_ReadSort(ap_id, origin, offset, data);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
origin	unsigned short	I	Reference value ELIB_ORG_SET (First data) ELIB_ORG_CUR (Current data) ELIB_ORG_END (Final data)
offset	Short	I	Search offset (*1) -n: Candidate n number before - : Candidate n number later
data	_ELIB_LIMIT_DATA *	O	Sort table read information (*2)
Return value	Type	I/O	Description
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark			

*1: The data moved by search offset is moved in units of telephone directories.

*2: Sort table read information

```
typedef struct ELIB_LIMIT_DATA
```

```
{
    unsigned short      object ;    /* Type: Object is used to determine whether to access
the following information */
    /* ELIB_PHONE (telephone directory)/ELIB_GROUP (telephone directory group) */
    union {
        /* Read information shared */
        _ELIB_MEDDATA    meddata ;/* Telephone directory data structure (*2-1) */
        _ELIB_GROUPDATA  group ;  /* Group name structure (*2-2) */
    } inf ;
} _ELIB_LIMIT_DATA ;
```

*2-1: Telephone directory data structure

For details about the telephone directory data structure, see [\[Memory Dial Read\]](#).

*2-2: Group name structure

```
typedef struct tagELIB_GROUPDATA {
    unsigned char        no ;    /* Telephone directory group number */
    _ELIB_GROUPNAME      data ;  /* Group name structure (*2-2-1) */
} _ELIB_GROUPDATA ;
```

*2-2-1: Group name structure

```
typedef struct _ELIB_GROUPNAME {
    unsigned char        g_name[ELIB_REF_KANJIMAX] ;    /* Kanji data */
    unsigned char        g_kana[ELIB_REF_KANAMAX ] ;    /* Read kana */
} _ELIB_GROUPNAME ;
#define ELIB_REF_KANJIMAX    22    /* Kanji data */
#define ELIB_REF_KANAMAX    22    /* Read
kana */
```

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- A temporary sort table must be created and the current location set as described in sort table creation ([\[Create Specified Function Sort Table\]](#) Elib_DEX_Limit_CreateSort()).

Note

- If task switching causes a different task to be used for the search, the temporary sort table will be changed. When the task is switched back to the original task and the result returned as the return value, the correct specified function information will not be able to be read. In this case, the calling side must use the original search conditions to reconfigure the temporary sort table. This case is a restriction of the data exchange service.

6.42 Count Specified Function Object From Search Table

Classification	Data exchange service support function		
Function	Count specified function object from search table	Symbol	Elib_DEX_Limit_CountObject
Functional overview	<p>- Returns the registration count specified by the collection count type from the temporary sort table.</p> <p>* UIM telephone directory data is not a target.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Limit_CountObject(ap_id, kind);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
kind	unsigned short	I	Collection count type ELIB_SEARCH_ALL: Telephone directory + telephone directory group count ELIB_SEARCH_PHONE: Telephone directory count ELIB_SEARCH_GROUP: Telephone directory group count ELIB_SEARCH_DIAL: Telephone number count ELIB_SEARCH_MAIL: Mail address count
Return value	Type	I/O	Description
Ret	int	O	Processing result Registration count (≥ 0): Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark	<p>Necessary procedure</p> <ul style="list-style-type: none"> - For each application, create a MSB object in advance using MsbObjectCreate.. - A temporary sort table must be created and the current location set as described in sort table creation ([Create Specified Function Sort Table] Elib_DEX_Limit_CreateSort()). <p>Note</p> <ul style="list-style-type: none"> - If task switching causes a different task to be used for the search, the temporary sort table will be changed. When the task is switched back to the original task and the result returned as the return value, the correct specified function information will not be able to be read. In this case, the calling side must use the original search conditions to reconfigure the temporary sort table. This case is a 		

restriction of the data exchange service.

6.43 Delete Single Data Item From Specified Function Sort Table

Classification	Data exchange service support function		
Function	Delete single data item from specified function sort table	Symbol	Elib_DEX_Limit_DelSort
Functional overview	<p>- Deletes the specified deletion data from the current data of the temporary sort table.</p> <p>- ELIB_DEL_PHONE (telephone directory data) can be used for deleting data even if the current location is a telephone directory group.</p> <p>* UIM telephone directory data is not a target.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Limit_DelSort(ap_id, kind, multino);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Kind	unsigned short	I	Deletion data information ELIB_DEL_PHONE: Telephone directory data ELIB_DEL_DIAL: Telephone number data ELIB_DEL_MAIL: Mail address data
multino	unsigned short	I	Deletion multi-number (*1) Valid range Telephone number: 0 to 3 Mail: 0 to 2
Return value	Type	I/O	Description
Ret	int	O	Processing result (*2) ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark			

***1: Deletion multi-number**

- If the deletion data information is telephone directory data, the deletion multi-number setting will be ignored.
- If the deletion data information is telephone number data, the valid range of the deletion multi-number setting is 0 to 3.
- If the deletion data information is mail address data, the valid range of the deletion multi-number setting is 0 to 2.

***2: Processing result**

- If the deletion multi-number setting is not within the valid range or data is not present at the specified location, an error will occur.
- If the current location is not set, there is no registration count, or telephone number/mail address/data of the specified multi-number cannot be found, an error will occur.
- If ELIB_DEL_DIAL (telephone number data) or ELIB_DEL_MAIL (mail address data) is not specified when the current location is a telephone directory group, an error will occur.

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- A temporary sort table must be created and the current location set as described in sort table creation ([\[Create Specified Function Sort Table\]](#) Elib_DEX_Limit_CreateSort()).

Note

- If task switching causes a different task to be used for the search, the temporary sort table will be changed. When the task is switched back to the original task and the result returned as the return value, the correct specified function information will not be able to be read. In this case, the calling side must use the original search conditions to reconfigure the temporary sort table. This case is a restriction of the data exchange service.

6.44 Request USIM Telephone Directory Structure Information

Classification	Data exchange service support function		
Function	Request USIM telephone directory structure information	Symbol	Elib_DEX_StructureInq
Functional overview			
<p>- Returns the USIM telephone directory structure (reusable, maximum registration count, name character string, etc.).</p>			
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_StructureInq (ap_id);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Return value	Type	I/O	Description
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error If an error occurs or a parameter is incorrect, event notification will not be performed.
Remark			
<p>- This interface is used to request collection of the UIM telephone directory structure. The result is posted by the event below after the request.</p>			
Table UIM telephone directory structure collection request notification result event format(_ELIB_DEX_EVT_STRUCTURE)			
Member name	Type	Description	
category	int	DataExchangeNotify	
subtype	int	DataExcNotify_StructureInq (Posts the result of the UIM telephone directory structure information collection request.)	
Info	int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure	
subinfo	int	Unused (always NULL)	
Data	_ELIB_STRUCTURE_INF	Telephone directory structure information structure (*1)	

* If the processing result is failure, the subinfo/data is invalid (NULL).

*1: Telephone directory structure information structure

```
typedef struct _ELIB_STRUCTURE_INF {  
    unsigned char pb_status :          /* Indicates the status of the UIM telephone directory
```



```

(*1-1)          */
unsigned char  pb_entrymax ;
               /* Maximum number of digits of UIM telephone directory registration count */
unsigned char  pb_maxlength ;
               /* Maximum number of digits of UIM telephone number registration */
unsigned char  pb_aimaxlength ;
               /* Maximum length of UIM name character string */
unsigned char  pb_snmaxentry;
               /* Maximum number of digits of UIM read kana registration count */
unsigned char  pb_snmaxlength;
               /* Maximum length of UIM read kana name character string */
unsigned char  pb_gnmaxentry ;
               /* Maximum number of characters of UIM group name */
unsigned char  pb_gnmaxlength;
               /* Maximum length of UIM group name character string */
unsigned char  pb_emailmaxentry ;
               /* Maximum number of digits of UIM mail address registration count */
unsigned char  pb_emailmaxlength ;
               /* Maximum length of UIM mail address character string */
} _ELIB_STRUCTURE_INF ;

```

*1-1: Information indicating whether the UIM telephone directory is available or not is set.

```

#define ELIB_TELBOOK_STOK    0x00    /* Telephone directory available */

```

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

6.45 Telephone Directory Group Name Request

Classification	Data exchange service support function		
Function	Telephone directory group name request	Symbol	Elib_DEX_GroupNameRef
Functional overview	<p>- Accesses the group name of the specified group number and the read kana of the group name.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_GroupNameRef(ap_id, g_no, g_name) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
g_no	unsigned char	I	Group number 0 to 19: Body 21 to 30: UIM * Group number 20 (UIM group 00) is not a target of group name editing. If specified, it will be handled as a parameter error.
G_name	_ELIB_GROUPNAME *	O	Group name (*1)
Return value	Type	I/O	Description
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMEERR: Parameter error
Remark	<p>*1: Group name structure</p> <pre>typedef struct _ELIB_GROUPNAME { unsigned char g_name[ELIB_REF_KANJIMAX] ; /* Kanji data */ unsigned char g_kana[ELIB_REF_KANAMAX] ; /* Read data */ }_ELIB_GROUPNAME; #define ELIB_REF_KANJIMAX 22 /* Kanji data */ #define ELIB_REF_KANAMAX 22 /* Read kana */</pre> <p>Necessary procedure</p> <p>- For each application, create a MSB object in advance using MsbObjectCreate..</p> <p>Note</p>		

6.46 Set Telephone Directory Group Name Request (Main Part)

Classification	Data exchange service support function		
Function	Set telephone directory group name request (main part)	Symbol	Elib_DEX_Mem_GroupNameSet
Functional overview	<p>- Sets the group name of the specified group number and the read kana of the group name.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Mem_GroupNameSet(ap_id, g_no, g_name);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
G_no	unsigned char	I	Group number 0 to 19: Body
g_name	_ELIB_GROUPNAME *	I	Group name (*1)
Return value	Type	I/O	Description
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark	<p>- If a null character string (the leading byte is 0x00) is passed to group name "g_name", it will be handled as group name deletion.</p> <p>*1: Group name structure</p> <pre>typedef struct _ELIB_GROUPNAME { unsigned char g_name[ELIB_REF_KANJIMAX] ; /* Kanji data */ unsigned char g_kana[ELIB_REF_KANAMAX] ; /* Read kana */ } _ELIB_GROUPNAME; #define ELIB_REF_KANJIMAX 22 /* Kanji data */ #define ELIB_REF_KANAMAX 22 /* Read kana */</pre> <p>Necessary procedure</p> <p>- For each application, create a MSB object in advance using MsbObjectCreate..</p> <p>Note</p>		

6.47 Set Telephone Directory Group Name Request (USIM)

Classification	Data exchange service support function		
Function	Set telephone directory group name request (USIM)	Symbol	Elib_DEX_Uim_GroupNameSet
Functional overview	<p>- Sets the group name of the specified group number and the read kana of the group name.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Uim_GroupNameSet(ap_id, g_no, g_name);		
Argument	Type	I/O	Description
Ap_id	unsigned int	I	Application ID
g_no	unsigned char	I	Group number 21 to 30: UIM * Group number 20 (UIM group 00) is not a target of group name editing. If specified, a parameter error will occur.
G_name	_ELIB_GROPNAME *	I	Group name (*1)
Return value	Type	I/O	Description
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMEERR: Parameter error If an error occurs or a parameter is incorrect, event notification will not be performed.
Remark			

- If a null character string (the leading byte is 0x00) is passed to group name "g_name", it will be handled as group name deletion.

*1: Group name structure

```
typedef struct _ELIB_GROUPNAME {
    unsigned char  g_name[ELIB_REF_KANJIMAX];    /* Kanji data      */
    unsigned char  g_kana[ELIB_REF_KANAMAX ];    /* Read data       */
}_ELIB_GROUPNAME;
#define ELIB_REF_KANJIMAX  22    /* Kanji data      */
#define ELIB_REF_KANAMAX   22    /* Read data       */
```

- This interface is used to request group name registration. The result is posted by the event below after the request.

Table 3.C-37.1 Telephone directory group name registration request result notification event format (`_ELIB_DEX_EVENT`)

Member name	Type	Description
category	int	DataExchangeNotify
subtype	int	DataExcNotify_UimGroupNameSetRes (Posts the result of the telephone directory group name registration request.)
Info	int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure
subinfo	int	Information indicating the data storage status (complete/incomplete) (*2) ELIB_DEX_COMP: Complete (All information is stored.) ELIB_DEX_INCOMP: Incomplete. Data has been dropped.
Data	int	Unused (always NULL)

*2: For UIM storage data, en-size kana characters are converted to em-size kana characters and registered. At this time, however, data can be dropped.

Necessary procedure

- For each application, create a MSB object in advance using `MsbObjectCreate..`
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

6.48 Copy USIM Telephone Directory Memory Request

Classification	Data exchange service support function																				
Function	Copy USIM telephone directory memory request	Symbol	Elib_DEX_Uim_MemoryCopyReq																		
Functional overview	<div>- Copies (cache) the USIM telephone directory data in the memory area.</div> <div>- Before this interface is executed, the 50 data items in the cache are overwritten. If there are not 50 items in the UIM, the empty records of the body are cleared.</div>																				
Include file	srv_dex.h																				
Calling sequence	int Elib_DEX_Uim_MemoryCopyReq (ap_id) ;																				
Argument	Type	I/O	Description																		
ap_id	unsigned int	I	Application ID																		
Return value	Type	I/O	Description																		
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error If an error occurs or a parameter is incorrect, event notification will not be performed.																		
Remark	<div>- This interface is used to request copying of the UIM telephone directory memory. The result is posted by the event below after the request.</div> <div>Table UIM telephone directory memory copy request result notification event format (_ELIB_DEX_EVENT)</div> <table><tr><th>Member name</th><th>Type</th><th>Description</th></tr><tr><td>category</td><td>int</td><td>DataExchangeNotify</td></tr><tr><td>subtype</td><td>int</td><td>DataExcNotify_UimMemoryCopyRes (Posts the result of the UIM telephone directory memory copu request.)</td></tr><tr><td>Info</td><td>int</td><td>Processing result ELIB_DEX_OK: Successful (complete) ELIB_DEX_NOK: Successful (Data has been dropped.) (*1) ELIB_DEX_NG: Failure</td></tr><tr><td>subinfo</td><td>int</td><td>≥0 (Number copied to memory) * Fixed to 0 at failure</td></tr><tr><td>Data</td><td>int</td><td>Unused (always NULL)</td></tr></table>			Member name	Type	Description	category	int	DataExchangeNotify	subtype	int	DataExcNotify_UimMemoryCopyRes (Posts the result of the UIM telephone directory memory copu request.)	Info	int	Processing result ELIB_DEX_OK: Successful (complete) ELIB_DEX_NOK: Successful (Data has been dropped.) (*1) ELIB_DEX_NG: Failure	subinfo	int	≥0 (Number copied to memory) * Fixed to 0 at failure	Data	int	Unused (always NULL)
Member name	Type	Description																			
category	int	DataExchangeNotify																			
subtype	int	DataExcNotify_UimMemoryCopyRes (Posts the result of the UIM telephone directory memory copu request.)																			
Info	int	Processing result ELIB_DEX_OK: Successful (complete) ELIB_DEX_NOK: Successful (Data has been dropped.) (*1) ELIB_DEX_NG: Failure																			
subinfo	int	≥0 (Number copied to memory) * Fixed to 0 at failure																			
Data	int	Unused (always NULL)																			

*1: ELIB_DEX_NOK: Successful (Data has been dropped.)
Currently, this status is not set.

* Even if there is no name in the UIM telephone directory data, the data is not disabled. The data is set in the name column in the following order of priority:
Furigana > Telephone number > Mail address
However, if there is no data for the three items above, en-size spaces are set in the name data.

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

6.49 Clear USIM Cache Data Request

Classification	Data exchange service support function		
Function	Clear USIM cache data request	Symbol	Elib_DEX_Uim_MemoryAllClear
Functional overview	<p>- Clears the entire UIM telephone directory area in the body memory.</p> <p>The data in the USIM card is not handled (in this case, clear).</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Uim_MemoryAllClear (void) ;		
Argument	Type	I/O	Description
None			
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK:Normal ELIB_DEX_NG:Abnormal
Remark	<p>Necessary procedure</p> <p>- For each application, create a MSB object in advance using MsbObjectCreate..</p> <p>Note</p>		

6.50 Check Telephone Directory Registration Data

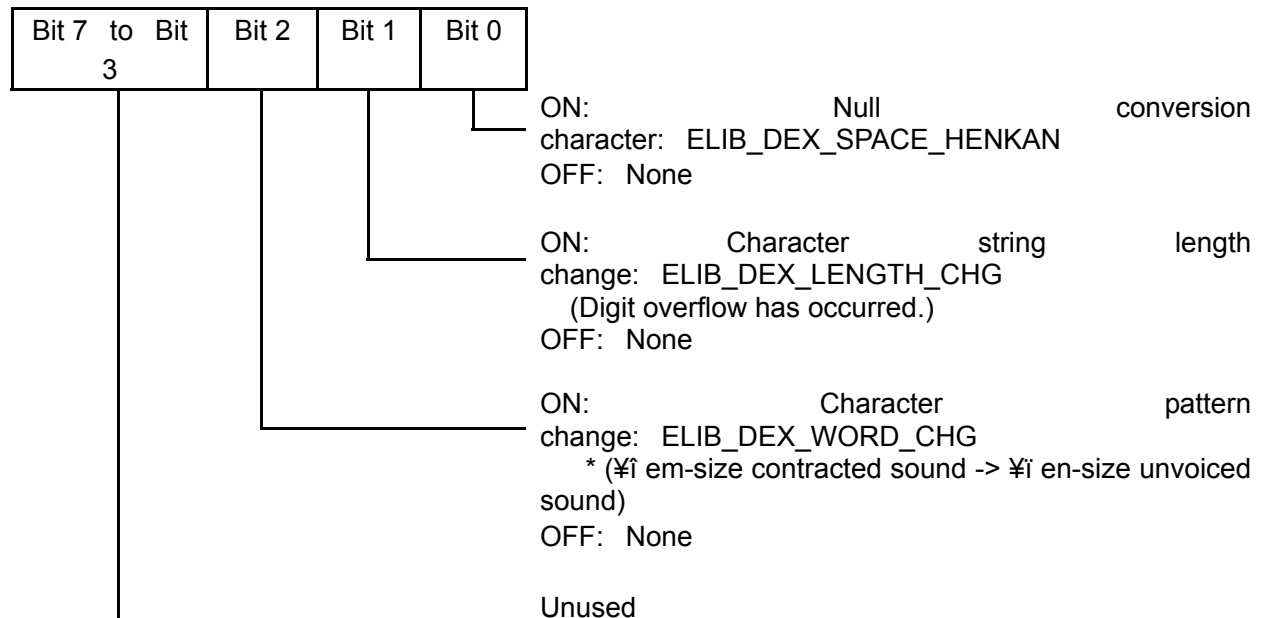
Classification	Data exchange service support function		
Function	Code check	Symbol	Elib_DEX_CodeCheck
Functional overview	<p>- Checks for the occurrence of data loss/digit overflow at code conversion during telephone directory data registration.</p> <p>- The following items are checked for the telephone directory data structure (_ELIB_MEDDATA).</p> <ul style="list-style-type: none"> - Name - Read kana - Telephone number - Mail address <p>- For the name and read kana, the check is performed after the family name and name are joined.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_CodeCheck(ap_id, stg_sts, data, sts);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
stg_sts	Int	I	Storage destination specification ELIB_DEX_MEMORY: Body ELIB_DEX_USIM: UIM
data	_ELIB_MEDDATA *	I	Telephone directory data structure (*1)
sts	unsigned char *	O	Check result (*2)
Return value	Type	I/O	Description
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark			

* 1:Telephone directory data structure

For details about the telephone directory data structure, see [\[Memory Dial Read\]](#).

*2: Check result

If data loss or digit overflow has not occurred, the result sts is set to ELIB_DEX_NO_CHG (0x00). In addition, if the result sts contains a value, the cause can be determined from the specifications below.



* Because there is a difference in the maximum number of registration digits between the telephone number (26 digits) of the body telephone directory and the telephone number (20 digits) of the UIM telephone directory, digits can be dropped when copying from the body to the UIM. If this case is detected, ELIB_DEX_LENGTH_CHG will be set on as the check result of this processing operation.

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..

6.51 Check Specified Ring Tone Setting

Classification	Data exchange service support function		
Function	Check specified ring tone setting	Symbol	Elib_DEX_LimitMelodyCheck
Functional overview	<p>- This function provides the processing necessary to determine whether a specific melody has been set or not set as a ring tone.</p> <p>* UIM telephone directory data is not a target.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_LimitMelodyCheck (ap_id, kind, ex_kind, flg);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
kind	unsigned long	I	Specified function type
ex_kind	unsigned long	I	Specified function type
flg	unsigned char *	O	Specified ring tone setting status (*1)
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark	<p>Necessary procedure</p> <p>- For each application, a MSB object must be created in advance using MsbObjectCreate.</p> <p>Note</p>		

6.52 Change Specified Ring Tone

Classification	Data exchange service support function		
Function	Change specified ring tone	Symbol	Elib_DEX_LimitMelodyChg
Functional overview	<p>- When the setting value of the specified handy function corresponding to the specified function conversion item (kind) is the conversion object specified ring tone, this function converts to the setting value (value) after conversion.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_LimitMelodyChg (ap_id, kind, ex_kind, flg, value);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
kind	unsigned long	I	Specified function conversion item
Ex_kind	unsigned long	I	Specified function conversion item
flg	unsigned char *	I	Conversion target original image specified bit string (*1)
value	Unsigned short	I	Setting value after conversion
Return value	Type	I/O	Description
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark	<p>Necessary procedure</p> <p>- For each application, a MSB object must be created in advance using MsbObjectCreate.</p> <p>Note</p>		

6.53 Auto Display Status Request

Classification	Data exchange service support function		
Function	Auto display status request	Symbol	Elib_DEX_AutoDial_GetReq
Functional overview	<p>- Returns the status of auto display.</p> <p>- If auto display has not been set, not set (ELIB_DEX_NOTSET) will be set.</p> <p style="padding-left: 20px;">* The auto_memno and multino values will be undefined.</p> <p>- Auto display is set/released as described in [C-14 Memory Dial Attribute Setting/Release].</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_AutoDial_GetReq (ap_id, auto_memno, multino) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
auto_memno	unsigned short *	O	Auto display memory number (0 to 699)
multino	unsigned char *	O	Multi-number (valid range: 0 to 3)
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_NOTSET: Not set ELIB_DEX_PRMERR: Parameter error
Remark	<p>Necessary procedure</p> <p>- For each application, a MSB object must be created in advance using MsbObjectCreate.</p> <p>Note</p>		

6.54 Memory Dial Total Deletion

Classification	Data exchange service support function		
Function	Memory dial total deletion	Symbol	Elib_DEX_DialAllClr
Functional overview	<p>- Clears all memory dials and the sort table.</p> <p>- Initializes the telephone directories, telephone numbers, mail addresses, URLs, addresses, memos, image information, movie information, telephone directory link information, registration count information, telephone directory update information, sort table information, and checksum information.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_DialAllClr (void) ;		
Argument	Type	I/O	Description
None			
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK :Normal ELIB_DEX_NG :Abnormal
Remark	<p>Necessary procedure</p> <p>Note</p>		

6.55 SMS Data Add/Update

Classification	Data exchange service support function		
Function	SMS data add/update	Symbol	Elib_DEX_Uim_SmsWrite
Functional overview	<p>- Specifies the record number of the SMS data and requests updating and registration of data for one SMS.</p> <p>- The processing result is provided at event notification.</p> <p>- In addition, the data format is based on the data format defined by the SMS service.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Uim_SmsWrite(ap_id, srec_no, data) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
srec_no	unsigned char	I	Record number of SMS data (>0)
data	_EUIM_SMSDATA *	I	SMS data to be registered (See Remark.)
Return value	Type	I/O	Description
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error If an error occurs or a parameter is incorrect, event notification will not be performed.
Remark			

- This interface is used to request SMS data registration to the USIM. The result is posted by the event below after the request

Table 3.D-1.1 SMS data addition/update result notification event format
(_ELIB_DEX_EVT_SMSWRITDEL)

Member name	Type	Description
category	int	DataExchangeNotify
subtype	int	DataExcNotify_SmsWriteRes (SMS data add/update response)
info	int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure
subinfo	int	Record number of status report
data	unsigned char	SMS status value after update Copy to the internal buffer and save/manage on the high-level side as required. (*1)

* If the processing result (info) is failure (ELIB_DEX_NG), the subinfo/data is invalid (NULL).

*1: Status value indicating the memory status)

```
#define EUIM_STS_NOREC      0x00      /* Unused record */
#define EUIM_STS_RCVSMS     0x01      /* Already read */
#define EUIM_STS_STILLRD    0x03      /* Not read */
#define EUIM_STS_SNDREQN    0x05      /* Sent (no report request) */
#define EUIM_STS_NSNDSMS    0x07      /* Not sent */
#define EUIM_STS_SNDREQY    0x0D      /* Sent (report requested/not received) */
#define EUIM_STS_SNDREQC    0x1D      /* Sent (report requested/received/stored in UIM) */
#define EUIM_STS_SNDREQS    0x15      /* Sent (report requested/received/not stored in UIM) */
```

* Reference: Memory status bit fields

B7	b6	b5	b4	b3	b2	b1	b0	
					*	*	0	Unused (empty)
					*	*	1	Used
				0	0	1		Receive mail (already read)
				0	1	1		Receive mail (not read)
				1	1	1		Send mail (not sent)
		*	*	1	0	1		Send mail (already sent)
		0	0	1	0	1		No status report request
		0	1	1	0	1		Wait for response after status report request
		1	0	1	0	1		Status report received, EF-SMSR not stored
		1	1	1	0	1		Status report received, EF-SMSR stored
								Unused (Spare: See 3G TS 31.101.)

[SMS data structure][I/-]
typedef struct
_EUIM_SMSDAT
A {
 unsigned


```

char          status;      /* SMS status value          *1 */
_ELIB_SMS_RP_ADDRESS  sc_address; /* Service center address*2 */
union tag {
    _ELIB_SMS_SUBMIT      snd_data; /* Send data          *3 */
    _ELIB_SMS_DELIVER      rcv_data; /* Receive data       *4 */
} sms_data;
_EUIM_SMSR          report; /* Status report (delivery notification) *5 */
} _EUIM_SMSDATA;

```

*2: Service center address

- For send data, the SC address stored using the SMS parameter is stored.
- For received data, the send source SC address collected from the receive data is stored.

*3: Send data structure

```

typedef struct tagELIB_SMS_SUBMIT {
    unsigned char    tp_msgInf; /* Message information */
    unsigned char    tp_mr; /* Message access information */
    _ELIB_SMS_TP_DA tp_da; /* Send destination address information */
    unsigned char    tp_pid; /* Protocol identifier */
    unsigned char    tp_dcs; /* Data coding scheme */
    unsigned char    tp_vp[ELIB_SMS_TPVP_MAX]; /* Send cycle information [7] */
    unsigned short   tp_udl; /* Use data length */
                                /* Data coding scheme: */
                                /* For GSM7bit, the data count is in units of 7 bits */
                                /* For UCS2, the data count is in units of 8 bits */
    unsigned char    tp_ud[ELIB_SMS_SUBMIT_MAX]; /* User data [140] */
                                /* Short message character string */
                                /* (The format depends on the data coding scheme */
} _ELIB_SMS_SUBMIT;

#define ELIB_SMS_SUBMIT_MAX (140)
#define ELIB_SMS_TPVP_MAX ( 7 )

```

* snd_data: Send data

*4: Receive data structure

```

typedef struct tagELIB_SMS_DELIVER {
    unsigned char    tp_msgInf; /* Mesage information */
    _ELIB_SMS_TP_DA tp_oa; /* Send source address information */
    unsigned char    tp_pid; /* Protocol identifier */
    unsigned char    tp_dcs; /* Data coding scheme */
    unsigned char    tp_scts[ELIB_SMS_TP_SCTS]; /* Service center timestamp [7] */
    unsigned char    tp_udl; /* User data length */
                                /* Data coding scheme: */
                                /* For GSM7bit, the data count is in units of 7 bits. */
}

```

```

/* For UCS2, the data count is in units of 8 bits. */
unsigned char    tp_ud[ELIB_SMS_DELIVER_MAX]; /* User data [140] */
/* Short message character string (The format depends on the data coding scheme */
} _ELIB_SMS_DELIVER;

#define ELIB_SMS_TP_SCTS    7
#define ELIB_SMS_DELIVER_MAX    (140)

* rcv_data: Receive data

*5: UIM storage status report

typedef struct _EUIM_SMSR {
    unsigned char    tp_msgInf; /* Message information */
    unsigned char    tp_mr; /* Message access information */
    _ELIB_SMS_TP_DA    tp_ra; /* Recipient address *7 */
    unsigned char    tp_scts[ELIB_SMS_TP_SCTS]; /* Service center timestamp */
    unsigned char    tp_dt[ELIB_SMS_TP_SCTS]; /* Send date and time */
    unsigned char    tp_st; /* Status information */
} _EUIM_SMSR;

#define ELIB_SMS_TP_SCTS    7

```

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

- This function does not convert codes. Therefore, for address data and user data, the character conversion library provided separately by the data exchange service must be used to convert and set the codes on the calling side in advance.

6.56 SMS Data Read

Classification	Data exchange service support function																				
Function	SMS data read	Symbol	Elib_DEX_Uim_SmsRead																		
Functional overview	<div>- Requests reading of one data item of the specified SMS memory number.</div> <div>- The processing result (read SMS data) is provided at event notification.</div>																				
Include file	Srv_dex.h																				
Calling sequence	int Elib_DEX_Uim_SmsRead(ap_id, srec_no) ;																				
Argument	Type	I/O	Description																		
ap_id	unsigned int	I	Application ID																		
srec_no	unsigned char	I	SMS data record number (>0)																		
Return value	Type	I/O	Description																		
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error If an error occurs or a parameter is incorrect, event notification will not be performed.																		
Remark	<div>- This interface is used to request reading of the SMS data in the UIM. The requested SMS data is posted by the event below.</div> <div>Table SMS data read result notification event format (_ELIB_DEX_EVT_SMSREAD)</div> <table><tr><th>Member name</th><th>Type</th><th>Description</th></tr><tr><td>category</td><td>int</td><td>DataExchangeNotify</td></tr><tr><td>subtype</td><td>int</td><td>DataExcNotify_SmsReadRes (SMS data read response)</td></tr><tr><td>Info</td><td>int</td><td>Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure ELIB_DEX_NOENTRY: No data</td></tr><tr><td>subinfo</td><td>int</td><td>Unused (always NULL)</td></tr><tr><td>Data</td><td>_EUIM_SMSDATA</td><td>SMS data of the specified record Copy to the internal buffer and save/manage on the high-level side as required. * For the data structure, see [SMS Data Add/Update].</td></tr></table> <div>* If the processing result (info) is failure (ELIB_DEX_NG) and there is no data (ELIB_DEX_NOENTRY), the subinfo/data is invalid (NULL).</div> <div>Necessary procedure<div>- For each application, create a MSB object in advance using MsbObjectCreate..</div></div>			Member name	Type	Description	category	int	DataExchangeNotify	subtype	int	DataExcNotify_SmsReadRes (SMS data read response)	Info	int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure ELIB_DEX_NOENTRY: No data	subinfo	int	Unused (always NULL)	Data	_EUIM_SMSDATA	SMS data of the specified record Copy to the internal buffer and save/manage on the high-level side as required. * For the data structure, see [SMS Data Add/Update] .
Member name	Type	Description																			
category	int	DataExchangeNotify																			
subtype	int	DataExcNotify_SmsReadRes (SMS data read response)																			
Info	int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure ELIB_DEX_NOENTRY: No data																			
subinfo	int	Unused (always NULL)																			
Data	_EUIM_SMSDATA	SMS data of the specified record Copy to the internal buffer and save/manage on the high-level side as required. * For the data structure, see [SMS Data Add/Update] .																			

- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

- This function does not convert codes. Therefore, this function also does not convert the codes for the address data and user data that are returned by the event to the calling side.

6.57 SMS Data Deletion

Classification	Data exchange service support function																				
Function	SMS data deletion	Symbol	Elib_DEX_Uim_SmsDelete																		
Functional overview	<div>- Requests deletion of one data item of the specified SMS memory number.</div> <div>- The processing result is provided at event notification.</div>																				
Include file	srv_dex.h																				
Calling sequence	int Elib_DEX_Uim_SmsDelete(ap_id, srec_no) ;																				
Argument	Type	I/O	Description																		
ap_id	Unsigned int	I	Application ID																		
srec_no	unsigned char	I	SMS data record number (>0)																		
Return value	Type	I/O	Description																		
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMEERR: Parameter error If an error occurs or a parameter is incorrect, event notification will not be performed.																		
Remark	<div>- This interface is used to request deletion of the SMS data in the UIM. The requested SMS data is posted by the event below.</div> <div>Table SMS data deletion request result notification event format (_ELIB_DEX_EVT_SMSWRITDEL)</div> <table><tr><th>Member name</th><th>Type</th><th>Description</th></tr><tr><td>category</td><td>Int</td><td>DataExchangeNotify</td></tr><tr><td>Subtype</td><td>Int</td><td>DataExcNotify_SmsDeleteRes (SMS data deletion response)</td></tr><tr><td>Info</td><td>Int</td><td>Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure</td></tr><tr><td>Subinfo</td><td>Int</td><td>Unused (always NULL)</td></tr><tr><td>Data</td><td>unsigned char</td><td>SMS data record number of the deletion record Copy to the internal buffer and save/manage on the high-level side as required. * For the data structure, see [SMS Data Add/Update].</td></tr></table> <div>* If the processing result (info) is failure (ELIB_DEX_NG), the subinfo/data is invalid (NULL).</div>			Member name	Type	Description	category	Int	DataExchangeNotify	Subtype	Int	DataExcNotify_SmsDeleteRes (SMS data deletion response)	Info	Int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure	Subinfo	Int	Unused (always NULL)	Data	unsigned char	SMS data record number of the deletion record Copy to the internal buffer and save/manage on the high-level side as required. * For the data structure, see [SMS Data Add/Update] .
Member name				Type	Description																
category	Int	DataExchangeNotify																			
Subtype	Int	DataExcNotify_SmsDeleteRes (SMS data deletion response)																			
Info	Int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure																			
Subinfo	Int	Unused (always NULL)																			
Data	unsigned char	SMS data record number of the deletion record Copy to the internal buffer and save/manage on the high-level side as required. * For the data structure, see [SMS Data Add/Update] .																			

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

6.58 SMS Parameter Setting

Classification	Data exchange service support function		
Function	SMS parameter setting	Symbol	Elib_DEX_Uim_SmsParamSet
Functional overview	<ul style="list-style-type: none"> - Updates the SMS parameter setting. - The processing result is provided at event notification. 		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Uim_SmsParamSet(ap_id, rec_no, parm) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
rec_no	Unsigned char	I	Record number (Always specify 1.)
parm	_EUIM_SMSPARM_INF *	I	SMS parameter data (See Remark.)
Return value	Type	I/O	Description
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark	<pre>typedef struct _EUIM_SMSPARM_INF { char alpha_id[ESMS_AID_SIZE]; /* Name (service center name, etc.) *1 */ unsigned char param_ind; /* Parameter indicator *2 */ _ELIB_SMS_TP_DA destination; /* Destination address *3 */ _ELIB_SMS_RP_ADDRESS service_center; /* Service center address *4 */ unsigned char protocol_id; /* Protocol identifier *5 */ unsigned char coding_scheme; /* Data coding scheme *6 */ unsigned char val_period; /* Validity period (0 to 255) *7 */ } _EUIM_SMSPARM_INF;</pre> <pre>#define ESMS_AID_SIZE (23) *1</pre> <p>*1: alpha_id:Alpha-Identifier An arbitrary SMS parameter title (name, etc.) can be set. The data is coded using GSM7 or UCS2. * If the data is coded using UCS2, ELIB_DEX_CODETYPE_UCS2 (0x80) is stored in the leading first byte. The name is stored from the second and subsequent bytes. * If the data is coded using GSM7, the name data is stored starting from the leading first byte. The area subsequent to the effective data (coded data) is padded using 0xFF. The length up</p>		

to 0xFF is judged to be the effective data length.

- Storage images of alpha_idMember of the _EUIM_SMSPARM_INF structure
UCS2 (When the effective data of UCS2 is 10 characters)

CC02 (which the effective data of CC02 is 16 characters)									
0	1	2	3	4	...	19	20	21	22
0x80	1st character		2nd character		...	10th character		0xFF	0xFF
↑ Byte 1 fixed to 0x80								↑ Bytes subsequent to effective data set to 0xFF.	

GSM7 (When the effective data of GSM7 is 20 characters)

0	1	2	3	...	19	20	21	22
1 st character	2nd character	3rd character	4th character	...	20th character	0xFF	0xFF	0xFF
↑ Character code entered from byte 1						↑ Bytes subsequent to effective data set to 0xFF.		

*2: param_ind:Parameter Indicators

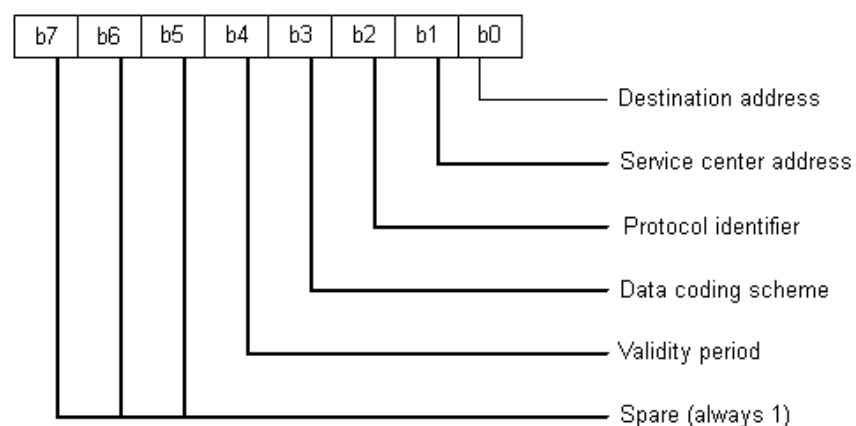
Bit management information indicating whether the information of each SMS parameter has been stored.

* Bit operations of this field: **0: Information, 1: No information**

(Reference: 3G TS 31.102 V3.2.0 4.2.27 EF-SMSP (Short message service parameters))

* Bit operations of this field at updating: **0: Update, 1: No update**

For example, to update only the center address data, set bit 1 center to 0 and the other bits to 1.



* Bit 0 is the LSB.

*3: destination:TP-Destination Address

Destination address

*4: service_center:TP-Service Center Address
Service center address

*5: protocol_id:TP-Protocol Identifier
Protocol identifier

*6: coding_scheme:TP-Data Coding Scheme
Data coding scheme

(7: val_period:TP-Validity Period
Validity period

TP-VP value	Explanation
0 to 143	(TP-VP+1) x 5 seconds
144 to 167	12 hours + ((TP-VP-143) x 30 seconds)
168 to 196	(TP-VP-166) x 1 day
197 to 255	(TP-VP-192) x 1 week

- This interface is used to request setting of the SMS data in the UIM. The requested SMS data is posted by the event below.

Table SMS parameter setting result notification event format (_ELIB_DEX_EVENT)

Member name	Type	Description
category	int	DataExchangeNotify
subtype	int	DataExcNotify_SmspSetRes (SMS parameter setting response)
Info	int	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Failure
subinfo	int	Record number (always 1)
Data	int	Unused (always NULL)

* If the processing result (info) is failure (ELIB_DEX_NG), the subinfo/data is invalid (NULL).

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

- This function does not convert codes. Therefore, for address data and name data, the character conversion library provided separately by the data exchange service must be used to convert and set the codes on the calling side in advance.

6.59 Get SMS Parameter Request

Classification	Data exchange service support function																				
Function	Get SMS parameter request	Symbol	Elib_DEX_Uim_SmsParamGet																		
Functional overview	<div>- Returns the SMS parameter setting information.</div> <div>- The processing result (SMS parameter information) is provided by event notification.</div>																				
Include file	srv_dex.h																				
Calling sequence	int Elib_DEX_Uim_SmsParamGet(ap_id, rec_no) ;																				
Argument	Type	I/O	Description																		
ap_id	unsigned int	I	Application ID																		
rec_no	unsigned char	I	Record number (Always specify 1.)																		
Return value	Type	I/O	Description																		
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error If an error occurs or a parameter is incorrect, event notification will not be performed.																		
Remark	<div>- This interface is used to request collection of the SMS data in the UIM. The requested SMS data is posted by the event below.</div> <div>Table 3.D-5.1 SMS parameter collection request result notification event format (_ELIB_DEX_EVT_SMSPARM)</div> <table><tr><th>Member name</th><th>Type</th><th>Description</th></tr><tr><td>category</td><td>int</td><td>DataExchangeNotify</td></tr><tr><td>subtype</td><td>int</td><td>DataExcNotify_SmspGetRes (SMS parameter collection response)</td></tr><tr><td>info</td><td>int</td><td>Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure</td></tr><tr><td>subinfo</td><td>int</td><td>Record number (always 1)</td></tr><tr><td>data</td><td>_EUIM_SMSPARM_INF</td><td>SMS parameter data Copy to the internal buffer and save/manage on the high-level side as required. * See [SMS parameter structure] in [SMS Parameter Setting].</td></tr></table> <div>* If the processing result (info) is failure (ELIB_DEX_NG), the subinfo/data is invalid (NULL).</div>			Member name	Type	Description	category	int	DataExchangeNotify	subtype	int	DataExcNotify_SmspGetRes (SMS parameter collection response)	info	int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure	subinfo	int	Record number (always 1)	data	_EUIM_SMSPARM_INF	SMS parameter data Copy to the internal buffer and save/manage on the high-level side as required. * See [SMS parameter structure] in [SMS Parameter Setting] .
Member name	Type	Description																			
category	int	DataExchangeNotify																			
subtype	int	DataExcNotify_SmspGetRes (SMS parameter collection response)																			
info	int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure																			
subinfo	int	Record number (always 1)																			
data	_EUIM_SMSPARM_INF	SMS parameter data Copy to the internal buffer and save/manage on the high-level side as required. * See [SMS parameter structure] in [SMS Parameter Setting] .																			

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

- This function does not convert codes. Therefore, this function also does not convert the codes for the address data and user data that are returned by the event to the calling side.

6.60 SMS Status Report Deletion

Classification	Data exchange service support function		
Function	SMS status report deletion	Symbol	Elib_DEX_Uim_SmsWrite_Del_Report
Functional overview	<p>- Specifies the record number of the SMS data and requests deletion of the delivery notification data for one SMS.</p> <p>* The SMS data format contents are the same as those defined using Elib_DEX_Uim_SmsWrite in [SMS Data Add/Update].</p>		
Include file	srv_dex.h		
Calling sequence	Int Elib_DEX_Uim_SmsWrite_Del_Report(ap_id, srec_no, data) ;		
Argument	Type	I/O	Description
ap_id	Unsigned int	I	Application ID
srec_no	unsigned char	I	SMS data record number (>0)
data	_EUIM_SMSDATA *	I	Target SMS data (See [SMS data] in [SMS Data Add/Update] .)
Return value	Type	I/O	Description
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error If an error occurs or a parameter is incorrect, event notification will not be performed.
Remark	<p>- This interface is used to request deletion of the status report information of the SMS data. The result is posted for the events after the request.</p> <p>- The processing result is provided by the same event as the notification event in [SMS Data Add/Update].</p> <p>Note: This interface cannot be executed records that do not contain any data.</p> <p>Necessary procedure</p> <p>- For each application, create a MSB object in advance using MsbObjectCreate..</p> <p>- Client needs to be registered at the event (see [Start Event Notification Request]).</p> <p>Note</p> <p>- This function does not convert codes. Therefore, for address data and user data, the character conversion library provided separately by the data exchange service must be used to convert and set the codes on the calling side in advance.</p>		

6.61 SMS Memory Capacity Flag Setting

Classification	Data exchange service support function																				
Function	SMS memory capacity flag setting	Symbol	Elib_DEX_Uim_SmssSet																		
Functional overview	<div>- Updates the SMS capacity flag (EF_SMSS) in the UIM that indicates the status of the memory area where the SMS receive data is stored.</div> <div>- If the receive SMS storage destination memory status (available/not available) is changed, the status must be posted to the NW together with using this interface to update the SMS capacity flag in the UIM.</div>																				
Include file	srv_dex.h																				
Calling sequence	int Elib_DEX_Uim_SmssSet(ap_id, mc_flg) ;																				
Argument	Type	I/O	Description																		
ap_id	unsigned int	I	Application ID																		
mc_flg	unsigned char	I	Memory available status (free status) ON: Available (memory free) OFF: Not available (memory full)																		
Return value	Type	I/O	Description																		
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error If an error occurs or a parameter is incorrect, event notification will not be performed.																		
Remark	<div>- This interface is used to request updating of the SMS capacity flag in the UIM. The requested SMS data is posted by the event below.</div> <div>Table SMS capacity flag update request result notification event format (_ELIB_DEX_EVENT)</div> <table><tr><th>Member name</th><th>Type</th><th>Description</th></tr><tr><td>category</td><td>int</td><td>DataExchangeNotify</td></tr><tr><td>subtype</td><td>int</td><td>DataExcNotify_SmssSetRes (SMS capacity flag setting response)</td></tr><tr><td>Info</td><td>int</td><td>Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure</td></tr><tr><td>subinfo</td><td>int</td><td>Unused (always NULL)</td></tr><tr><td>Data</td><td>int</td><td>Unused (always NULL)</td></tr></table>			Member name	Type	Description	category	int	DataExchangeNotify	subtype	int	DataExcNotify_SmssSetRes (SMS capacity flag setting response)	Info	int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure	subinfo	int	Unused (always NULL)	Data	int	Unused (always NULL)
Member name	Type	Description																			
category	int	DataExchangeNotify																			
subtype	int	DataExcNotify_SmssSetRes (SMS capacity flag setting response)																			
Info	int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure																			
subinfo	int	Unused (always NULL)																			
Data	int	Unused (always NULL)																			

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

6.62 Get SMS Memory Capacity Flag Request

Classification	Data exchange service support function																				
Function	SMS memory capacity flag access	Symbol	Elib_DEX_Uim_SmssGet																		
Functional overview	<p>- Returns the value of the SMS capacity flag in the USIM.</p> <p>See the previous section [SMS Memory Capacity Flag Setting].</p>																				
Include file	srv_dex.h																				
Calling sequence	int Elib_DEX_Uim_SmssGet(ap_id) ;																				
Argument	Type	I/O	Description																		
ap_id	unsigned int	I	Application ID																		
Return value	Type	I/O	Description																		
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error If an error occurs or a parameter is incorrect, event notification will not be performed.																		
Remark	<p>- This interface is used to request access of the SMS capacity flag in the UIM. The requested SMS data is posted by the event below.</p> <p>Table SMS capacity flag access request result notification event format (_ELIB_DEX_EVENT)</p> <table><tr><th>Member name</th><th>Type</th><th>Description</th></tr><tr><td>category</td><td>Int</td><td>DataExchangeNotify</td></tr><tr><td>subtype</td><td>Int</td><td>DataExcNotify_SmssGetRes (SMS capacity flag access response)</td></tr><tr><td>Info</td><td>Int</td><td>Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure</td></tr><tr><td>subinfo</td><td>Int</td><td>Memory available status (free status) ON: Available (memory free) OFF: Not available (memory full)</td></tr><tr><td>Data</td><td>int</td><td>Unused (always NULL)</td></tr></table> <p>* If the processing result (info) is failure (ELIB_DEX_NG), the subinfo/data is invalid (NULL).</p>			Member name	Type	Description	category	Int	DataExchangeNotify	subtype	Int	DataExcNotify_SmssGetRes (SMS capacity flag access response)	Info	Int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure	subinfo	Int	Memory available status (free status) ON: Available (memory free) OFF: Not available (memory full)	Data	int	Unused (always NULL)
Member name	Type	Description																			
category	Int	DataExchangeNotify																			
subtype	Int	DataExcNotify_SmssGetRes (SMS capacity flag access response)																			
Info	Int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure																			
subinfo	Int	Memory available status (free status) ON: Available (memory free) OFF: Not available (memory full)																			
Data	int	Unused (always NULL)																			
Necessary procedure																					

- For each application, create a MSB object in advance using MsbObjectCreate..
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

6.63 Send APDU Request

Classification	Data exchange service support function																	
Function	Send APDU request	Symbol	Elib_DEX_CDF_APDU_Req															
Functional overview	<div>- Requests to send an APDU command to USIM_TAF</div>																	
Include file	srv_dex.h tafsvc.h (USIM_TAF distribution header)																	
Calling sequence	int Elib_DEX_CDF_APDU_Req(ap_id, req_add);																	
Argument	Type	I/O	Description															
ap_id	unsigned int	I	Application ID															
req_add	T_CDF_APDU_REQ *	I	Pointer to the APDU issuing request data structure (*1) (*2)															
Return value	Type	I/O																
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error ELIB_DEX_BUSY: Processing in progress * If an APDU issuing request for USIM_TAF is in progress, processing in progress will be returned and the request to USIM_TAF will not be executed.															
Remark	<div>- This interface is used to request APDU issuing to the UIM. The requested APDU issuing request response data is posted by the event below.</div> <div>Table APDU issuing result response event format (_ELIB_DEX_EVT_APDU)</div> <table><tr><th>Member name</th><th>Type</th><th>Description</th></tr><tr><td>category</td><td>Int</td><td>DataExchangeNotify</td></tr><tr><td>subtype</td><td>Int</td><td>DataExcNotify_CDF_APDU_Res (APDU issuing result response)</td></tr><tr><td>Info</td><td>Int</td><td>Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure</td></tr><tr><td>subinfo</td><td>Int</td><td>Unused (always NULL)</td></tr></table>			Member name	Type	Description	category	Int	DataExchangeNotify	subtype	Int	DataExcNotify_CDF_APDU_Res (APDU issuing result response)	Info	Int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure	subinfo	Int	Unused (always NULL)
Member name				Type	Description													
category	Int	DataExchangeNotify																
subtype	Int	DataExcNotify_CDF_APDU_Res (APDU issuing result response)																
Info	Int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure																
subinfo	Int	Unused (always NULL)																

Data	T_CDF_APDU_RES	APDU issuing request response data structure (*1) (*3)
------	----------------	--

* If the processing result (info) is failure (ELIB_DEX_NG), the subinfo/data is invalid (NULL).

(*1)

- For the APDU issuing request data and response data, the data exchange service does not check the data contents or convert the data.

(*2)

[APDU issuing request data structure]

typedef struct tag_T_CDF_APDU_REQ

```
{
    unsigned char          ain_flg;          /**< AIN enable/disable
    flag                    */
    unsigned char          ain_data;         /**< AIN                      */
    unsigned char          file_sel_info[8]; /**< File selection
    information             */
    unsigned char          command_cnt;      /**< Valid command count      */
    T_CDF_APDU_COMMAND     command[6];     /**< Command list          (*4) */
}T_CDF_APDU_REQ;
```

(*3)

[APDU issuing request response data structure]

typedef struct tag_T_CDF_APDU_RES

```
{
    unsigned char          comp_flg;        /**< AIN return/"GG" return flag */
    unsigned char          ain_data;        /**< AIN                      */
    unsigned char          file_sel_info[8]; /**< File selection
    information             */
    unsigned char          result_cnt;      /**< Command result count      */
    T_CDF_APDU_RESULT      result[6];     /**< Command result          (*4) */
}T_CDF_APDU_RES;
```

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

- When using this function, the header file of USIM_TAF distribution must be included.

6.64 PUBKEY Read Request

Classification	Data exchange service support function		
Function	PUBKEY read request	Symbol	Elib_DEX_CDF_PUBKEY_Req
Functional overview			
<div>- Requests reading of the PUBKEY to USIM_TAF.</div>			
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_CDF_PUBKEY_Req(ap_id, req_msg) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Req_msg	_ELIB_DEX_CDF_PUBKEY_REQ_MSG *	I	Pointer to the PUBKEY read request structure (*1)
Return value	Type	I/O	
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error ELIB_DEX_BUSY: Processing in progress * If a PUBKEY read request for USIM_TAF is in progress, processing in progress will be returned and the request to USIM_TAF will not be executed.
Remark			

- This interface is used to request reading of the PUBKEY to the UIM. The requested PUBKEY read result response data is posted by the event below.

Table PUBKEY read result response event format (_ELIB_DEX_EVT_PUBKEY)

Member name	Type	Description
category	int	DataExchangeNotify
subtype	int	DataExcNotify_CDF_PUBKEY_Res (PUBKEY read result response)

Info	int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure ELIB_DEX_PRMERR: Parameter error
subinfo	int	Unused (always NULL)
Data	ELIB_DEX_CDF_PUBKEY_RES	PUBKEY read result structure (*2)

* If the processing result (info) is failure (ELIB_DEX_NG), the subinfo/data is invalid (NULL).

(*1)

[PUBKEY read request structure]

```
typedef struct _ELIB_DEX_CDF_PUBKEY_REQ_MSG {
    unsigned char    df_ind[4];        /* DF_ID      */
    unsigned char    ef_ind[2];        /* EF_ID      */
} _ELIB_DEX_CDF_PUBKEY_REQ_MSG ;
```

(*2)

[PUBKEY read result structure]

```
typedef struct _ELIB_DEX_CDF_PUBKEY_RES {
    unsigned char    pub_key[128];     /* PUB_KEY    */
    unsigned char    exp_value[3];     /* EXP_VALUE  */
    unsigned char    mac_data[20];     /* MAC        */
} _ELIB_DEX_CDF_PUBKEY_RES ;
```

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

6.65 Digital Signature Operation Execution Request

Classification	Data exchange service support function		
Function	Digital signature operation execution request	Symbol	Elib_DEX_CDF_SIGNATURE_Req
Functional overview	<p>- Requests digital signature operation execution to USIM_TAF.</p> <p>- Before this function is called, the high-level APPLICATION must collect the PIN2 input remaining count (the authentication service provides the interface) and check whether the PIN2 has been locked. If the PIN2 has been locked, this function cannot be used to request digital signature operation execution. (If this function is called when the PIN2 is locked, ELIB_DEX_DATA_NG(PIN2NG) will be returned for INFO of the event information.)</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_CDF_SIGNATURE_Req(ap_id, req_msg) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
req_msg	_ELIB_DEX_CDF_SIGNATURE_REQ_MSG *	I	Pointer to the digital signature operation execution request structure (*1) (*1)
Return value	Type	I/O	
ret	Int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMMERR: Parameter error ELIB_DEX_BUSY: Processing in progress * If a digital signature operation execution request for USIM_TAF is in progress, processing in progress will be returned and the request to USIM_TAF will not be executed.
Remark			

(*1)

[Digital signature operation execution request structure]

```
typedef struct _ELIB_DEX_CDF_SIGNATURE_REQ_MSG {
    unsigned char    df_ind[4] ;           /* FID */
    unsigned char    pin2_data[8] ;       /* PIN2 */
    unsigned char    digest_len ;         /* Digest data length */
    unsigned char    digest_data[127] ;   /* Digest data */
} _ELIB_DEX_CDF_SIGNATURE_REQ_MSG ;
```

* For the entered PIN2, the high-level APPLICATION must check the number of significant digits and effective characters (value).

* The PIN2 value must be left-justified and the unused free area padded with 0xFF.

Example: For PIN2 data "1234"

```
pin2_data : { 0x31, 0x32, 0x33, 0x34, 0xFF, 0xFF, 0xFF, 0xFF }
```

* The digest area must be left-justified, the length made effective, and the free area padded with 0x00.

- This interface is used to request digital signature operation execution to the UIM. The requested digital signature operation execution result response data is posted by the event below.

Table Digital signature operation execution result response event format
(_ELIB_DEX_EVT_SIGNATURE)

Member name	Type	Description
Category	int	DataExchangeNotify
subtype	int	DataExcNotify_CDF_SIGNATURE_Res (Digital signature operation execution result response)
Info	int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_DATA_NG: PIN2 NG ELIB_DEX_NG: Failure ELIB_DEX_PRMERR: Parameter error
subinfo	int	PIN2 input remaining count 0 to 2 * Valid only when the processing result is PIN2NG(ELIB_DEX_DATA_NG). The PIN2 is locked at input remaining count 0.
Data	_ELIB_DEX_CDF_SIGNATURE_RES	Digital signature operation execution result structure (*2)

* If the processing result (info) is failure (ELIB_DEX_NG), the subinfo/data is invalid (NULL).

(*2)

[Digital signature operation execution result structure]

```
typedef struct _ELIB_DEX_CDF_SIGNATURE_RES {
    unsigned char    signature_len ;      /* Signature data length */
    unsigned char    signature[128] ;    /* Signature data */
} _ELIB_DEX_CDF_SIGNATURE_RES ;
```

* The signature data is left-justified, the length made effective, and the free area padded with 0x00.

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

6.66 UIM Certificate Read Request

Classification	Data exchange service support function		
Function	UIM certificate read request	Symbol	Elib_DEX_CDF_EF_Read_Req
Functional overview	<p>- Reads the nonvolatile UIM certificate file cache data or the UIM certificate file of the UIM (uses the USIM_TAF interface). When the UIM certificate file is read from the UIM, the read information is expanded nonvolatily.</p> <p>* If all of the UIM certificate file information (EF_ACTIVATE, EF_ROOT, EF_SUBROOT, and EF_CERT) read from the UIM is successfully written to nonvolatile memory, the UIM certificate cache enable/disable flag is set to enable.</p> <p>* Error handling (Conditions for updating the UIM certificate cache enable/disable flag)</p> <ul style="list-style-type: none"> - If any one of the data items fails to be written when writing the UIM certificate file information (EF_ACTIVATE, EF_ROOT, EF_SUBROOT, and EF_CERT) read from the UIM to nonvolatile memory, the UIM certificate cache enable/disable flag is set to disable. - If any one of the data items of the UIM certificate file information (EF_ACTIVATE, EF_ROOT, EF_SUBROOT, and EF_CERT) cached in nonvolatile memory fails to be read, the UIM certificate cache file information is collected again from the UIM. 		
Include file	srv_dex.h tafsvc.h (USIM_TAF distribution header)		
Calling sequence	int Elib_DEX_CDF_EF_Read_Req(ap_id, mode) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
mode	int	I	Processing mode ELIB_DEX_CDF_READ: UIM certificate regular read (at DB construction) If the nonvolatile cache flag is enable, the certificate file is read from the nonvolatile memory. If the nonvolatile cache flag is disable, the certificate file is read from the UIM. ELIB_DEX_CDF_UIM_READ: UIM certificate forced read (at DB reconstruction) The certificate file is read from the UIM regardless of the

			nonvolatile cache flag enable/disable setting.
Return value	Type	I/O	
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error ELIB_DEX_BUSY: Processing in progress * If a UIM certificate read request for USIM_TAF is in progress, processing in progress will be returned and the request to USIM_TAF will not be executed.

Remark

- This interface is used to request reading of the UIM certificate to the UIM. The requested UIM certificate read result response data is posted by the event below.

Table UIM certificate read result response event format (_ELIB_DEX_EVT_EFREAD)

Member name	Type	Description
category	int	DataExchangeNotify
subtype	int	DataExcNotify_CDF_EF_Read_Res (UIM certificate read result response)
Info	int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure
subinfo	int	Unused (always NULL)
Data	T_CDF_EF_READ_DATA	UIM certificate read result storage data structure (*1)

* If the processing result (info) is failure (ELIB_DEX_NG), the subinfo/data is invalid (NULL).

(*1)

- For the UIM certificate data, the data exchange service does not check the data contents or convert the data.

Necessary procedure

- For each application, a MSB object must be created in advance using MsbObjectCreate.
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

- When using this function, the header file of USIM_TAF distribution must be included.

6.67 UIM Certificate Enable/Disable Setting Request

Classification	Data exchange service support function		
Function	UIM enable/disable request	certificate setting	Symbol Elib_DEX_CDF_Activate_Req
Functional overview	<p>- Requests UIM certificate disable/enable to USIM_TAF.</p> <p>- Updates the nonvolatile relevant information when processing terminates normally.</p> <p>* When the USIM-TAF processing result is successful</p> <p>The USIM-TAF processing result is set in the processing result flag of the nonvolatile relevant information (EF_ACTIVATE). The UIM certificate cache enable/disable flag is not updated (because the UIM certificate file information items other than EF_ACTIVATE are not always enable).</p> <p>* When the USIM-TAF processing result is other than successful</p> <p>The USIM-TAF processing result is set in the processing result flag of the nonvolatile relevant information (EF_ACTIVATE). The UIM certificate cache enable/disable flag is updated to disable.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_CDF_Activate_Req(ap_id, req_msg);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
req_msg	_ELIB_DEX_CDF_ACTIVATE_REQ_MSG *	I	Pointer to the UIM certificate enable/disable request structure (*1)
Return value	Type	I/O	
ret	Int	O	<p>Processing result</p> <p>ELIB_DEX_OK: Normal</p> <p>ELIB_DEX_NG: Abnormal</p> <p>ELIB_DEX_PRMMERR: Parameter error</p> <p>ELIB_DEX_BUSY: Processing in progress</p> <p>* If a UIM certificate enable/disable request for USIM_TAF is in progress, processing in progress will be returned and the request to USIM_TAF will not be executed.</p>

Remark

(*1)

[UIM certificate enable/disable request structure]

```
typedef struct _ELIB_DEX_CDF_ACTIVATE_REQ_MSG {
    unsigned char    activate[2];    /* EF_ACTIVATE contents */
} _ELIB_DEX_CDF_ACTIVATE_REQ_MSG ;
```

- This interface is used to request UIM certificate enable/disable to the UIM. The requested UIM certificate enable/disable result response data is posted by the event below.

Table UIM certificate enable/disable request result response event format (_ELIB_DEX_EVENT)

Member name	Type	Description
category	int	DataExchangeNotify
subtype	int	DataExcNotify_CDF_Activate_Res (UIM certificate enable/disable request result response)
Info	int	Processing result ELIB_DEX_OK: Successful ELIB_DEX_NG: Failure ELIB_DEX_PRMERR: Parameter error
subinfo	int	Unused (always NULL)
Data	int	Unused (always NULL)

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..
- Client needs to be registered at the event (see [\[Start Event Notification Request\]](#)).

Note

6.68 UIM Cache Disable Request

Classification	Data exchange service support function		
Function	UIM cache disable request	Symbol	Elib_DEX_CDF_Cache_Off
Functional overview	<p>- Disables the UIM certificate enable/disable information managed in nonvolatile memory.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_CDF_Cache_Off(ap_id) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Return value	Type	I/O	
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark	<p>Necessary procedure</p> <p>- For each application, a MSB object must be created in advance using MsbObjectCreate.</p> <p>Note</p>		

6.69 UIM Certificate Cache Data Access

Classification	Data exchange service support function		
Function	UIM certificate cache data access	Symbol	Elib_DEX_CDF_Cache_Read
Functional overview	<p>- Reads the UIM certificate information managed in nonvolatile memory.</p> <p>- To collect the information, specify the data type of the information to be read. For a high-level APPLICATION, required data read area must be allocated based on the data type.</p>		
Include file	srv_dex.h tafsvc.h (USIM_TAF distribution header)		
Calling sequence	int Elib_DEX_CDF_Cache_Read(ap_id, kind, data) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
kind	int	I	Data type (*1)
data	void *	O	Pointer to the data read structure (*2)
Return value	Type	I/O	
ret	int	O	Processing result ELIB_DEX_OK: Normal ELIB_DEX_NG: Abnormal ELIB_DEX_PRMERR: Parameter error
Remark	<p>(*1)</p> <p>This function can read the following five information items:</p> <p>(1) ELIB_DEX_CDF_ONOFF: UIM certificate cache enable/disable information</p> <p>(2) ELIB_DEX_CDF_EF_ACTIVATE: EF_ACTIVATE cache</p> <p>(3) ELIB_DEX_CDF_EF_ROOT: EF_ROOT cache</p> <p>(4) ELIB_DEX_CDF_EF_SUBROOT: EF_SUBROOT cache</p> <p>(5) ELIB_DEX_CDF_EF_CERT: EF_CERT cache</p> <p>* If the (1) UIM certificate cache enable/disable information is disable, the data of (2) to (5) is disabled. To access the data of (2) to (5), first access (1) to check that the value of the UIM certificate cache enable/disable information is set to enable. Then, specify (2) to (5) and call this function.</p> <p>(*2)</p> <p>List of data read structures used by this function</p> <p>(1) UIM certificate cache enable/disable information</p> <pre>typedef struct _ELIB_DEX_CDF_EF_CACHE_TYPE {</pre>		

```

        unsigned char    cert_cause ;           /* Cache enable/disable */
    } _ELIB_DEX_CDF_EF_CACHE_TYPE ;

```

(2) EF_ACTIVATE data structure

```

typedef struct _ELIB_DEX_CDF_EF_ACTIVATE_TYPE {
    unsigned char    activate_cause ;           /* EF_ACTIVATE read result */
    unsigned char    activate[2] ;             /* EF_ACTIVATE contents */
} _ELIB_DEX_CDF_EF_ACTIVATE_TYPE ;

```

(3) EF_ROOT data structure

```

typedef struct _ELIB_DEX_CDF_EF_ROOT_TYPE {
    unsigned char    root_cause ;               /* EF_ROOT read result */
    unsigned char    root[1500] ;              /* EF_ROOT contents */
} _ELIB_DEX_CDF_EF_ROOT_TYPE ;

```

(4) EF_SUBROOT data structure

```

typedef struct _ELIB_DEX_CDF_EF_SUBROOT_TYPE {
    unsigned char    subroot_cause ;           /* EF_SUBROOT read result */
    unsigned char    subroot[1500] ;          /* EF_SUBROOT contents */
} _ELIB_DEX_CDF_EF_SUBROOT_TYPE ;

```

(5) EF_CERT data structure

```

typedef struct _ELIB_DEX_CDF_EF_CERT_TYPE {
    unsigned char    cert_cause ;              /* EF_CERT read result */
    unsigned char    cert[1500] ;             /* EF_CERT contents */
} _ELIB_DEX_CDF_EF_CERT_TYPE ;

```

Necessary procedure

- For each application, create a MSB object in advance using MsbObjectCreate..

Note

- For the file read result, the USIM_TAF distribution header file must be included at access.
- The data exchange service handles management (setting) of the nonvolatile data. A setting interface is not provided for high-level applications.

6.70 History Registration

Classification	Data exchange service support function		
Function	History registration	Symbol	Elib_DEX_Hist_Write
Functional overview	<p>- Registers the incoming/outgoing history.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Hist_Write(handle, kind, data);		
Argument	Type	I/O	Description
handle	unsigned short	I	Handle number (fixed from the request source) ELIB_DEX_HDL_USERM (User main) ELIB_DEX_HDL_USERS (User sub)
kind	unsigned short	I	History type ELIB_DEX_HIST_RCV (Incoming history) ELIB_DEX_HIST_SEND (Outgoing history) ELIB_DEX_HIST_KONZAI (Mixed outgoing history)
data	_ELIB_HISTDATA *	I	History data storage address
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal end ELIB_DEX_NG: Abnormal end ELIB_DEX_PRMERR: Parameter error
Remark			

- Indicates the history data storage structure.

```
typedef struct tagELIB_HISTDATA {      Incoming/outgoing history information structure

    unsigned char    Dial[ELIB_SIPMAX+1] ;      /* Dial number          */
    unsigned char    Yomi[ELIB_KANAMAX+1] ;     /* Read data            */
    unsigned char    Char[ELIB_KANJIMAX+1];     /* Name data            */
    unsigned char    Dummy1[3];                 /* Spare 1              */
    _ELIB_CAL_DATETIME DateTime ;              /* Outgoing/incoming date and
time      */

    unsigned char    Flag ;                     /* History flag information      *1   */
    unsigned char    Cause ;                    /* Non-notification reason      *2   */
    unsigned short   Icon ;                     /* Icon                        */
    unsigned char    HistKind;                  /* Communication type          *3   */
    unsigned char    Notice;                    /* Additional number information *4   */
    unsigned char    CallTime;                  /* Call time                   *5   */
    unsigned char    Connection;                /* Connection time             *6   */
    unsigned short   SndRcvModelIcon;           /* Outgoing/incoming mode icon */
    unsigned char    Unuse_CallTime; /* Call time disabled incoming 1: Call time disabled */
    unsigned char    Dummy2;                    /* Spare 2                    */
} _ELIB_HISTDATA;

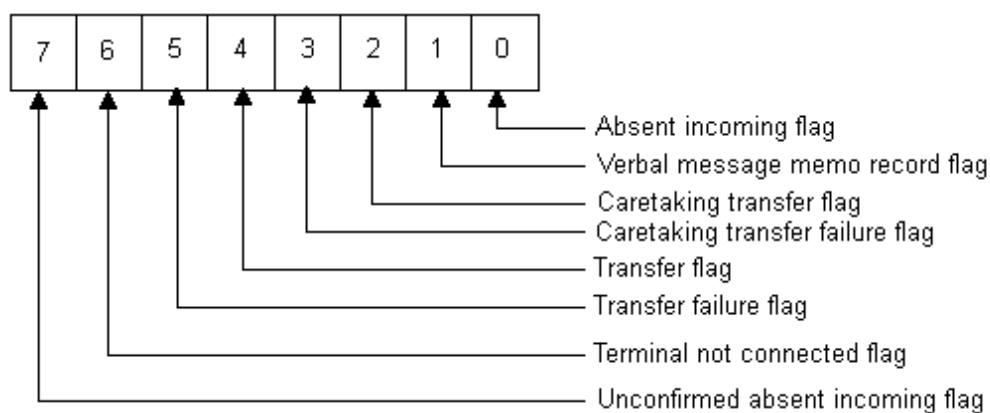
typedef struct tagELIB_CAL_DATETIME {      Date and time information structure

    unsigned short   Year;                      /* Year                      */
    unsigned char    Mon;                       /* Month (1 to 12)           */
    unsigned char    Day;                       /* Day (1 to 31)             */
    unsigned char    Week;                      /* Day of the week (0: Sunday to 6: Saturday) */
    unsigned char    Hour;                      /* Hour (0 to 23)            */
    unsigned char    Min;                       /* Minute (0 to 59)          */
    unsigned char    Sec;                       /* Second (0 to 59)          */
} _ELIB_CAL_DATETIME;

#define ELIB_SIPMAX      50    /* Maximum number of dial number digits */
#define ELIB_KANAMAX     32    /* Maximum number of read kana characters */
#define ELIB_KANJIMAX    32    /* Maximum number of name characters */
```

*1: History flag information details of the incoming/outgoing history information structure

1 byte



iãSymbol definitions>

ELIB_DEX_ATR_FUZA	0x0001	Absent incoming flag
ELIB_DEX_ATR_MEMO	0x0002	Verbal message memo record flag
ELIB_DEX_ATR_RUSU	0x0004	Caretaking transfer flag
ELIB_DEX_ATR_RUSU_NG	0x0008	Caretaking transfer failure flag
ELIB_DEX_ATR_TENSO	0x0010	Transfer flag
ELIB_DEX_ATR_TENSO_NG	0x0020	Transfer failure flag
ELIB_DEX_ATR_NO_TERM	0x0040	Terminal not connected flag
ELIB_DEX_ATR_NOCHK	0x0080	Unconfirmed absent incoming flag
ELIB_DEX_ATR_FLAG	0x00ff	History flag information mask bit

- The maximum number of bytes is 50 (ELIB_SIPMAX) for the dial number, 32 (ELIB_KANAMAX) for the read data, and 32 (ELIB_KANJIMAX) for the name data. A terminating character is set for the final byte. If any of the maximum sizes is exceeded, the character in the final byte will be ignored. The data will be registered with the final byte replaced with the terminating character.

*2: For the history flag information and non-notification reason, the value specification is valid only for incoming history. For other histories, the value is ignored even if specified and 0x00 is registered.

*3: The following values are set for the communication type.

ELIB_DEX_KIND_FOMA	(FOMA)
ELIB_DEX_KIND_PACKET	(Packet)
ELIB_DEX_KIND_64K	(64K data communication)
ELIB_DEX_KIND_AV	(External AV)
ELIB_DEX_KIND_TVTEL	(Videophone)
ELIB_DEX_KIND_REMOTE	(Remote monitoring)

*4: The following values are set for the additional number information.

ELIB_DEX_NOTICE_ON	(Originating number notification setting)
ELIB_DEX_NOTICE_OFF	(Originating number non-notification setting)
ELIB_DEX_NOTICE_NOSET	(No originating number setting)

*5: For the call time, the value specification is valid only for incoming history. Because the range is not checked, it must be checked in the high-level process.

*6: The following values are set for the connection information.

ELIB_DEX_CONNECTION_OK	(Connection successful (conversation))
ELIB_DEX_CONNECTION_NG	(Connection unsuccessful (absent incoming/charges 0))
ELIB_DEX_CONNECTION_OBSCURE	(Connection obscure)

- For the outgoing history data, the same dial is not recorded more than once. If a registration attempt is made using the same dial as that of data already recorded (regardless of character/no character), the latest data will be registered as the outgoing history. The older data will be deleted.

- The history count is checked for each history type and communication type. If any of the maximum counts below is exceeded, the oldest history corresponding to each pattern will be deleted and the history registered.

The patterns for each history type and communication type and the maximum counts are as follows.

Incoming history (FOMA, videophone, and remote monitoring): 30 maximum
 Incoming history (packet, 64K data communication, and external AV): 30 maximum
 Outgoing history (FOMA, videophone, and remote monitoring): 30 maximum
 Mixed outgoing history (FOMA, videophone, and remote monitoring): 30 maximum
 Mixed outgoing history (packet, 64K data communication, and external AV): 30 maximum

- If this function is called after the search table has been created (after calling the Elib_DEX_Hist_Open function), close processing will be called within this function and the search table will be deleted.

6.71 History Access Start

Classification	Data exchange service support function		
Function	History access start	Symbol	Elib_DEX_Hist_Open
Functional overview	<p>- Enables use of the specified history type.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Hist_Open (handle, kind, filter, value);		
Argument	Type	I/O	Description
handle	unsigned short	I	Handle number (fixed from the request source) ELIB_DEX_HDL_USERM (User main) ELIB_DEX_HDL_USERS (User sub)
kind	unsigned short	I	History type ELIB_DEX_HIST_RCV (Incoming history) ELIB_DEX_HIST_SEND (Outgoing history) ELIB_DEX_HIST_KONZAI (Mixed outgoing history)
filter	unsigned short	I	Filter information See note *1 in the Remark column.
value	unsigned short	I	Filter value See note *1 in the Remark column.
Return value	Type	I/O	
ret	int	O	Processing result ELIB_DEX_OK: Normal end ELIB_DEX_NG: Abnormal end ELIB_DEX_PRMERR: Parameter error
Remark			

- This function creates a search table based on the specified search conditions.
- When the search table is created, the current location is set at the beginning.
- Different handle numbers can be used to create search tables of the same history type to create multiple search tables.
- When using the same handle number to create a search table of the same history type, the search table of the current history type is deleted before the search table is created.
- The same handle number cannot be used to create a search table of multiple history types. When a search table of a different history type is created, the current history type is forcibly deleted. The search table is then created using the new history type.

*1: Setting filter information and filter values enables the collection of history data of history flag information equivalent to using history access (ELIB_DEX_Hist_Read). The values below can be set for the filter information and filter values. These values can be combined by logical add.

iãSymbol definitions>

ELIB_DEX_ATR_FUZA	0x0001	Absent incoming flag
ELIB_DEX_ATR_MEMO	0x0002	Verbal message memo record flag
ELIB_DEX_ATR_RUSU	0x0004	Caretaking transfer flag
ELIB_DEX_ATR_RUSU_NG	0x0008	Caretaking transfer failure flag
ELIB_DEX_ATR_TENSO	0x0010	Transfer flag
ELIB_DEX_ATR_TENSO_NG	0x0020	Transfer failure flag
ELIB_DEX_ATR_NO_TERM	0x0040	Terminal not connected flag
ELIB_DEX_ATR_NOCHK	0x0080	Unconfirmed absent incoming flag
ELIB_DEX_ATR_FLAG	0x00ff	History flag information mask bit

*** If all are targets, 0 is set.**

- The following history data is included in the incoming history.

(FOMA, packet, 64K data communication, external AV, videophone, and remote monitoring)

- The following history data is included in the outgoing history.
(FOMA, videophone, and remote monitoring)
- The following history data is included in the mixed incoming history.
(FOMA, packet, 64K data communication, external AV, videophone, and remote monitoring)

6.72 History Access

Classification	Data exchange service support function		
Function	History access	Symbol	Elib_DEX_Hist_Read
Functional overview	<p>- Accesses the incoming/outgoing history.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Hist_Read(handle, kind, origin, offset, data, cnt);		
Argument	Type	I/O	Description
handle	unsigned short	I	Handle number (fixed from the request source) ELIB_DEX_HDL_USERM (User main) ELIB_DEX_HDL_USERS (User sub)
kind	unsigned short	I	History type ELIB_DEX_HIST_RCV (Incoming history) ELIB_DEX_HIST_SEND (Outgoing history) ELIB_DEX_HIST_KONZAI (Mixed outgoing history)
origin	unsigned short	I	Access reference position ELIB_ORG_SET (First) ELIB_ORG_CUR (Current) ELIB_ORG_END (Final)
offset	short	I	Move offset -n: n files before the origin O: File specified by origin n: n files after the origin
data	_ELIB_HISTDATA *	O	History data storage address * To read multiple data items, the calling side must allocate area for the number of data items.
cnt	unsigned char	I	Read count (*1)
Return value	Type	I/O	
ret	int	O	Processing result ELIB_DEX_OK: Normal end ELIB_DEX_NG: Abnormal end ELIB_DEX_PRMERR: Parameter error ELIB_DEX_NOENTRY: Cannot be found
Remark			

- To use this function, history access start (Elib_DEX_Hist_Open) must have been performed using a handle number.
- After execution of this function, the current location is moved to the next location of the access data if multiple count has been specified for the read count. If 1 has been specified for the read count, the access location becomes the current location.
- Indicates the history data storage structure.

```
typedef struct tagELIB_HISTDATA {      Incoming/outgoing history information structure
    unsigned char      Dial[ELIB_SIPMAX+1]          /* Dial number */
    unsigned char      Yomi[ELIB_KANAMAX+1];        /* Read data */
    unsigned char      Char[ELIB_KANJIMAX+1];        /* Name data */
    unsigned char      Dummy1[3];                   /* Spare 1 */
    _ELIB_CAL_DATETIME DateTime;                    /* Outgoing/incoming date and time */
    unsigned char      Flag;                         /* History flag
information          *2 */
    unsigned char      Cause;                       /* Non-notification reason */
    unsigned short     Icon;                        /* Icon */
    unsigned char      HistKind;                    /* Communication
type          *3 */
    unsigned char      Notice;                      /* Additional number information */
    unsigned char      CallTime;                    /* Call
time          *4 */
    unsigned char      Connection;                  /* Connection
information          *5 */
    unsigned short     SndRcvModelcon;              /* Outgoing/incoming mode icon */
    unsigned char      Unuse_CallTime;              /* Call time disabled incoming 1: Call time
disabled */
    unsigned char      Dummy2;                      /* Spare 2 */
} _ELIB_HISTDATA;
```

```

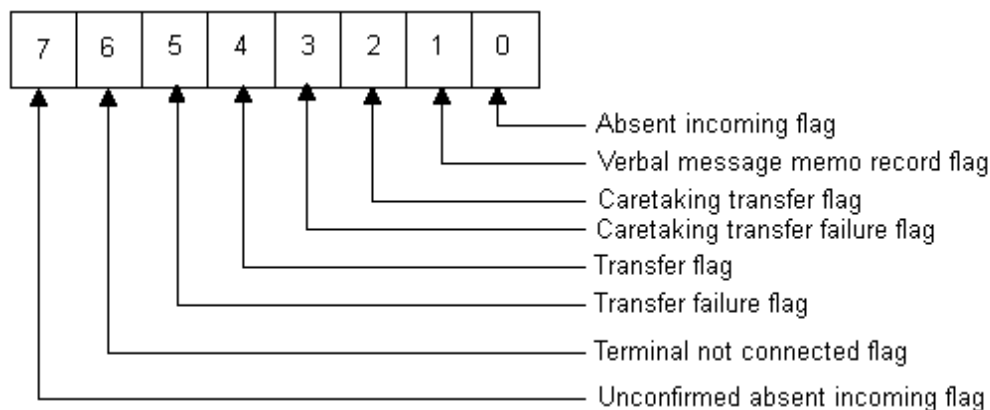
typedef struct tagELIB_CAL_DATETIME {    Date and time information structure
    unsigned short    Year;        /* Year */
    unsigned char     Mon;        /* Month (1 to 12) */
    unsigned char     Day;        /* Day (1 to 31) */
    unsigned char     Week;       /* Day of the week (0: Sunday to 6: Saturday) */
    unsigned char     Hour;       /* Hour (0 to 23) */
    unsigned char     Min;        /* Minute (0 to 59) */
    unsigned char     Sec;        /* Second (0 to 59) */
} _ELIB_CAL_DATETIME;

#define ELIB_SIPMAX      50        /* Maximum number of dial number digits */
#define ELIB_KANAMAX     32        /* Maximum number of read kana characters */
#define ELIB_KANJIMAX    32        /* Maximum number of name characters */

```

*2: Indicates the history flag information details of the incoming/outgoing history information structure.

1 byte



<Symbol definitions>

ELIB_DEX_ATR_FUZAI	0x0001	Absent incoming flag
ELIB_DEX_ATR_MEMO	0x0002	Verbal message memo record flag
ELIB_DEX_ATR_RUSU	0x0004	Caretaking transfer flag
ELIB_DEX_ATR_RUSU_NG	0x0008	Caretaking transfer failure flag
ELIB_DEX_ATR_TENSO	0x0010	Transfer flag
ELIB_DEX_ATR_TENSO_NG	0x0020	Transfer failure flag
ELIB_DEX_ATR_NO_TERM	0x0040	Terminal not connected flag
ELIB_DEX_ATR_NOCHK	0x0080	Unconfirmed absent incoming flag
ELIB_DEX_ATR_FLAG	0x00ff	History flag information mask bit

Setting the filter information and filter values at ELIB_DEX_Hist_Open enables the collection data to be selected from this history flag information. A logical conjunction of the history flag information and filter information of the history data is made. If the value matches the filter

- The maximum number of bytes of 50 (ELIB_SIPMAX), 32 (ELIB_KANAMAX), and 32 (ELIB_KANJIMAX) are returned for the dial number, read data, and name data. A terminating character is returned for the final byte.

*1: The following values are set for the read count.

For incoming history: 1 to 60

For outgoing history: 1 to 30

For mixed outgoing history: 1 to 60

*3: The following values are set for the communication type.

ELIB_DEX_KIND_FOMA	(FOMA)
ELIB_DEX_KIND_PACKET	(Packet)
ELIB_DEX_KIND_64K	(64K data communication)
ELIB_DEX_KIND_AV	(External AV)
ELIB_DEX_KIND_TVTEL	(Videophone)
ELIB_DEX_KIND_REMOTE	(Remote monitoring)

*4: The call time is valid only for incoming history.

*5: The following values are set for the connection information.

ELIB_DEX_CONNECTION_OK	(Connection successful (conversation))
ELIB_DEX_CONNECTION_NG	(Connection unsuccessful (absent incoming/charges 0))
ELIB_DEX_CONNECTION_OBSCURE	(Connection obscure)

Note at single data item access

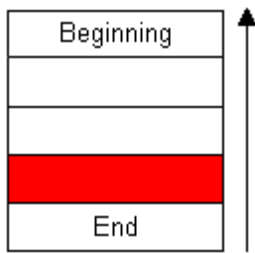
When this processing operation is executed, the current location is obtained by adding the access reference location and the move offset location. The data at that location is read. When processing is completed, note that the current location is not moved to the next data location.

The histories are registered starting from the beginning in order of older history first. Therefore, if ELIB_ORG_SET is specified for the reference location and 0 for the move offset, the oldest history will be collected. See the example figures below.

Example: Collecting one data item from the end of the incoming history

Access reference location: End, Move offset: -1, Read count: 1

The painted location is accessed.



Note at multiple data item collection

At multiple read, the offset count amount is obtained from the access reference location.

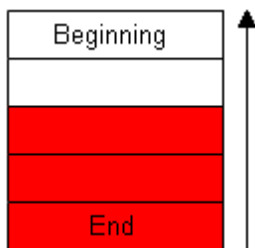
See the example figure below.

Collecting three data items

Example 1: Collecting three data items from the end of the incoming history

Access reference location: End, Move offset: -1, Read count: 3

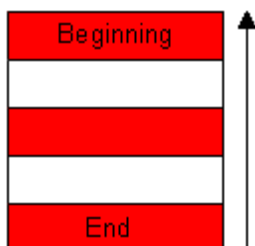
The painted locations are accessed.



Example 2: Collecting three data items from the end of the incoming history

Access reference location: End, Move offset: -2, Read count: 3

The painted locations are accessed.



6.73 History Deletion

Classification	Data exchange service support function		
Function	History deletion	Symbol	Elib_DEX_Hist_Del
Functional overview	<p>- Deletes the incoming/outgoing history of the current location.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Hist_Del (handle);		
Argument	Type	I/O	Description
handle	unsigned short	I	Handle number (fixed from the request source) ELIB_DEX_HDL_USERM (User main) ELIB_DEX_HDL_USERS (User sub)
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal end ELIB_DEX_NG: Abnormal end ELIB_DEX_PRMERR: Parameter error
Remark	<p>- After this processing operation is executed, the current location indicates the next data (if there is no next data, the current location is not set).</p> <p>- To use this function, history access start (Elib_DEX_Hist_Open) must have been performed using a handle number.</p>		

6.74 Delete Unconfirmed Absent Incoming Flag Request

Classification	Data exchange service support function		
Function	Delete unconfirmed absent incoming flag request	Symbol	Elib_DEX_NoChkFlg_Del
Functional overview	<p>- Deletes the unconfirmed absent incoming flag set in the history flag information of the current location.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_NoChkFlg_Del(handle);		
Argument	Type	I/O	Description
Handle	unsigned short	I	Handle number (fixed from the request source) ELIB_DEX_HDL_USERM (User main) ELIB_DEX_HDL_USERS (User sub)
Return value	Type	I/O	Description
Ret	int	O	Processing result ELIB_DEX_OK: Normal end ELIB_DEX_NG: Abnormal end ELIB_DEX_PRMEERR: Parameter error
Remark	<p>- To use this function, history access start (Elib_DEX_Hist_Open) must have been performed using a handle number.</p>		

6.75 History Access Stop

Classification	Data exchange service support function		
Function	History access stop	Symbol	Elib_DEX_Hist_Close
Functional overview	<p>- Stops use of the history type currently in use.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Hist_Close (handle);		
Argument	Type	I/O	Description
handle	unsigned short	I	Handle number (fixed from the request source) ELIB_DEX_HDL_USERM (User main) ELIB_DEX_HDL_USERS (User sub)
Return value	Type	I/O	
ret	int	O	Processing result ELIB_DEX_OK: Normal end ELIB_DEX_NG: Abnormal end ELIB_DEX_PRMERR: Parameter error
Remark	<p>- If end processing has been performed for files for which end processing is performed, normal end will be returned.</p> <p>- End processing deletes the search table and releases the current information.</p>		

6.76 Total Deletion by History Type

Classification	Data exchange service support function		
Function	Total deletion by history type	Symbol	Elib_DEX_Hist_AllDel
Functional overview	<p>- Completely deletes the incoming/outgoing history of the specified history type.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Hist_AllDel(kind);		
Argument	Type	I/O	Description
Kind	unsigned short	I	History type ELIB_DEX_HIST_RCV (Incoming history) ELIB_DEX_HIST_SEND (Outgoing history) ELIB_DEX_HIST_KONZAI (Mixed outgoing history)
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal end ELIB_DEX_NG: Abnormal end ELIB_DEX_PRMERR: parameter error
Remark	<p>- If total deletion is performed for a history type of a created search table (after the Elib_DEX_Hist_Open function is called), that history type will be forcibly closed and ELIB_DEX_NG returned for all subsequent history access requests.</p>		

6.77 History Count Request

Classification	Data exchange service support function		
Function	History count request	Symbol	Elib_DEX_Hist_Cnt
Functional overview	<p>- Returns the incoming/outgoing history count.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Hist_Cnt(handle, kind, data);		
Argument	Type	I/O	Description
handle	unsigned short	I	Handle number (fixed from the request source) ELIB_DEX_HDL_USERM (User main) ELIB_DEX_HDL_USERS (User sub)
kind	unsigned short	I	History type ELIB_DEX_HIST_RCV (Incoming history) ELIB_DEX_HIST_SEND (Outgoing history) ELIB_DEX_HIST_KONZAI (Mixed outgoing history)
data	_ELIB_HISTCNT *	O	History count storage address
Return value	Type	I/O	Description
ret	Int	O	Processing result ELIB_DEX_OK: Normal end ELIB_DEX_NG: Abnormal end ELIB_DEX_PRMERR: Parameter error
Remark	<pre>typedef struct tagELIB_HISTCNT { unsigned char hist1 unsigned char hist2 } _ELIB_HISTCNT ;</pre> <p>Incoming/outgoing history count structure Incoming/outgoing count Absent incoming count</p> <p>- If the history type is outgoing history, the absent incoming count of the incoming/outgoing history count structure is not used.</p> <p>- If this function is called for search table created (after the Elib_DEX_Hist_Open function is called), close processing will be called within this function and the search table deleted.</p>		

6.78 Unconfirmed Absent Incoming History Count Request

Classification	Data exchange service support function		
Function	Unconfirmed absent incoming history count request	Symbol	Elib_DEX_Hist_NoChkCnt
Functional overview	<p>- Returns the unconfirmed absent incoming history count.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_Hist_NoChkCnt (handle, cnt);		
Argument	Type	I/O	Description
handle	unsigned short	I	Handle number (fixed from the request source) ELIB_DEX_HDL_USERM (User main) ELIB_DEX_HDL_USERS (User sub)
cnt	int *	O	Unconfirmed absent incoming count
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal end ELIB_DEX_NG: Abnormal end ELIB_DEX_PRMERR: Parameter error
Remark	<p>- If this function is called for search table created (after the Elib_DEX_Hist_Open function is called), close processing will be called within this function and the search table deleted.</p>		

6.79 Address History Setting Access

Classification	Data exchange service support function		
Function	Address history setting access	Symbol	Elib_DEX_MailHistSetRef
Functional overview	<p>- Accesses the setting of the send address history/receive address history.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_MailHistSetRef (mode, type, offset, HistData);		
Argument	Type	I/O	Description
Mode	int	I	Mode ELIB_DEX_FUNCSET: Set ELIB_DEX_FUNCREF: Access
Type	unsigned short	I	History type ELIB_DEX_MAILHIST_RCV: Receive address history ELIB_DEX_MAILHIST_TRM: Send address history
Offset	unsigned short	I	Setting access offset value 0 to 29 (Set the offset value from the oldest data.) This is not used if mode is set.
HistData	_ELIB_DEX_MAILHIST *	I/O	Address history information storage area
Return value	Type	I/O	Description
Ret	int	O	Processing result ELIB_DEX_OK: Normal end ELIB_DEX_NG: Abnormal end ELIB_DEX_PRMERR: Parameter error ELIB_DEX_NOENTRY: Cannot be found
Remark			

Address history data structure

```
typedef struct tagELIB_DEX_MAILHIST
```

```
{
    unsigned char    Address[ELIB_MAILMAX+1];    /* Address data */
    unsigned char    Name[ELIB_KANJIMAX+1];      /* Name data */
    unsigned char    Kana[ELIB_KANAMAX+1];       /* Furigana data */
    unsigned char    Attrib;                     /* Mail attributes */
    _ELIB_CAL_DATETIME DateTime;                 /* Send/receive date and time */
    unsigned char    Result;                     /* Send result */
    unsigned char    Cause;                      /* Non-notification reason */
    unsigned short   Icon;                       /* Icon */
    unsigned char    ReStatus;                   /* Can be returned/cannot be returned */
    unsigned char    FSecurity;                  /* Folder security */
}_ELIB_DEX_MAILHIST;
```

```
typedef struct tagELIB_CAL_DATETIME {    Date and time information structure
```

```
    unsigned short   Year;                     /* Year */
    unsigned char    Mon;                       /* Month (1 to 12) */
    unsigned char    Day;                       /* Day (1 to 31) */
    unsigned char    Week;                      /* Day of the week (0: Sunday to 6: Saturday) */
    unsigned char    Hour;                      /* Hour (0 to 23) */
    unsigned char    Min;                       /* Minute (0 to 59) */
    unsigned char    Sec;                       /* Second (0 to 59) */
}_ELIB_CAL_DATETIME;
```

```
#define ELIB_MAILMAX    50    /* Maximum number of mail address digits */
#define ELIB_KANJIMAX   32    /* Maximum number of name data characters */
#define ELIB_KANAMAX    32    /* Maximum number of furigana data characters */
```

Note

- For the history data, the same address is not registered more than once. If an attempt is made to register the same address, the latest data will be registered as the history and the older data deleted.
- For the send/receive history, the address history setting order is sorted in order of keys.

6.80 Address History Deletion

Classification	Data exchange service support function		
Function	Address history deletion	Symbol	Elib_DEX_MailHistErase
Functional overview	<p>- Deletes one send address history/receive address history item.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_MailHistErase (type, offset);		
Argument	Type	I/O	Description
type	unsigned short	I	History type ELIB_DEX_MAILHIST_RCV: Receive address history ELIB_DEX_MAILHIST_TRM: Send address history
offset	unsigned short	I	Deletion offset value 0 to 29 (Set the offset value from the oldest data.) Handle number (fixed from the request source)
Return value	Type	I/O	Description
ret	Int	O	Processing result ELIB_DEX_OK: Normal end ELIB_DEX_NG: Abnormal end ELIB_DEX_PRMERR: Parameter error ELIB_DEX_NOENTRY: Cannot be found
Remark			

6.81 Address History Total Deletion

Classification	Data exchange service support function		
Function	Address history total deletion	Symbol	Elib_DEX_MailHistEraseAll
Functional overview	<p>- Deletes all send address history/receive address history items.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_MailHistEraseAll (type);		
Argument	Type	I/O	Description
type	unsigned short	I	History type ELIB_DEX_MAILHIST_RCV: Receive address history ELIB_DEX_MAILHIST_TRM: Send address history
Return value	Type	I/O	Description
ret	int	O	Processing result ELIB_DEX_OK: Normal end ELIB_DEX_NG: Abnormal end ELIB_DEX_PRMERR: Parameter error
Remark			

6.82 Address History Registration Count Request

Classification	Data exchange service support function		
Function	Address history registration request	Symbol	Elib_DEX_MailHistCnt
Functional overview	<p>- Returns the send address history/receive address history registration count.</p>		
Include file	srv_dex.h		
Calling sequence	int Elib_DEX_MailHistCnt (type);		
Argument	Type	I/O	Description
type	unsigned short	I	History type ELIB_DEX_MAILHIST_RCV: Receive address history ELIB_DEX_MAILHIST_TRM: Send address history
Return value	Type	I/O	Description
ret	int	O	Processing result Registration count (0 to 30): Normal end ELIB_DEX_NG: Abnormal end ELIB_DEX_PRMERR: Parameter error
Remark			

7. Record/Playback

7.1 Record/Playback Event Notification Request Stop

7-1 Record/Playback Event Notification Request Stop			
Category	Record/Playback service support function		
Description	Stops the record/playback event notification request.		
Name	Elib_REC_Cancel		
Prototype	int Elib_REC_Cancel (unsigned int ap_id, int event) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Event	int	I	Event (See Functional description <Notification events> in A-1 Record/playback Event Notification Request Start.)
Return value	Type	I/O	Description
Ret	int	O	Normal : ELIB_REC_OK Abnormal : ELIB_REC_NG Incorrect argument value : ELIB_REC_NG_PARM
Include file	srv_rec.h		
Functional description	Releases notification to the application of the record/playback event registered using the Elib_REC_Request() function.		
Related message			
Pre-conditions			
Note			
Constraints			
Usage	<pre> unsigned int ap_id ; /* Application ID */ int event ; /* Event to be deleted */ int ret ; /* Function return value */ /** Deletion parameter setting **/ ap_id = AM_AP_ID_JAM_MAIN ; event = RecordNotify_RPSTOP ; /* Record stop notification */ ret = Elib_REC_Cancel (ap_id, event) ; /* Record/playback event notification request stop */ </pre>		

7.2 Record Data information Collection Request

7-2 Record Data Information Collection Request			
Category	Record/playback service support function		
Description	Requests collection of the record data information.		
Name	Elib_REC_DataInfo_Get		
Prototype	Int Elib_REC_DataInfo_Get (unsigned int ap_id, _Elib_REC_PARAM *rec_param, _Elib_REC_STATUS *status) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
rec_param	_Elib_REC_PARAM *	I	Specify the data to be collected. <pre> typedef struct tag_Elib_REC_PARAM { int path ; /* For this function, the path value is invalid. */ int mode ; /* Record/playback mode */ /* The following values can be set. */ /* ELIB_REC_DEN : Verbal message memo */ /* ELIB_REC_VOICE : Voice memo */ int memo_no ; /* Record/playback data number */ /* The following values can be set. */ /* ELIB_REC_MEMO1 : Voice memo */ /* ELIB_REC_TELMSG1 : Verbal message memo 1 */ /* ELIB_REC_TELMSG2 : Verbal message memo 2 */ /* ELIB_REC_TELMSG3 : Verbal message memo 3 */ /* ELIB_REC_TELMSG4 : Verbal message memo 4 */ /* ELIB_REC_TELMSG5 : Verbal message memo 5 */ } _Elib_REC_PARAM ; </pre> * When voice memo has been specified for mode, this

			function returns ELIB_REC_NG_PARM if verbal message memo 1 to 5 has been specified for memo_no. If verbal message memo has been specified for mode, this function returns ELIB_REC_NG_PARM even if a voice memo value has been specified for memo_no.
Status	_Elib_REC_STATUS *	O	<p>Voice data record status data: See the Functional description.</p> <p>* If the return value is other than normal end (ELIB_REC_OK), the value of this argument will be undefined (unpredictable).</p>
Return value	Type	I/O	Description
Ret	int	O	<p>Normal : ELIB_REC_OK</p> <p>Abnormal : ELIB_REC_NG</p> <p>Incorrect argument value : ELIB_REC_NG_PARM</p> <p>No record status data : ELIB_REC_NODATA</p> <p>Recording or deleting : ELIB_REC_DISCORD</p>
Include file	srv_rec.h		
Functional description	<p>- Collects the record status of the specified voice data.</p> <p>- If this function is used while record data is being deleted, ELIB_REC_DISCORD will be returned as the return value.</p> <p>The following information can be collected.</p> <pre>typedef struct tag_Elib_REC_STATUS { unsigned char rec_status ; /* Record status */ /* 0: No recording */ /* 1: Recording */ unsigned char month ; /* Month */ unsigned char day ; /* Day */ unsigned char hour ; /* Hour */ unsigned char min ; /* Minute */ unsigned char week ; /* Day of week */ /* 0 : Sunday */ /* 1 : Monday */ /* 2 : Tuesday */ /* 3 : Wednesday */ /* 4 : Thursday */ /* 5 : Friday */ /* 6 : Saturday */ }</pre>		

	<pre> unsigned char no_info ; /* Non-notification incoming information Invalid for voice memo */ /* ELIB_REC_CAUSE_NONE No non-notification reason */ /* ELIB_REC_CAUSE_SECRET User notification refuse */ /* ELIB_REC_CAUSE_COMPETE Service competition */ /* ELIB_REC_CAUSE_PUBLIC Public telephone */ /* ELIB_REC_CAUSE_NOSERVICE Service distribution not permitted */ unsigned char dial_buf[ELIB_REC_DIALMAX] ; /* Dial buffer Invalid for voice memo */ unsigned char dial_cnt ; /* Dial counter Invalid for voice memo */ unsigned char word_buf[ELIB_REC_KANJIMAX] ; /* Character buffer Invalid for voice memo */ unsigned char word_cnt ; /* Character counter Invalid for voice memo */ unsigned char btyomi[ELIB_REC_KANAMAX] ; /* En-size read data Invalid for voice memo */ unsigned char btyomilen ; /* En-size read data length Invalid for voice memo */ unsigned char grp_num ; /* Group number Invalid for voice memo */ int icon ; /* Multi-icon Invalid for voice memo */ int cvtv_flg ; /* At voice/TV communication (*1) */ /* ELIB_REC_ROKUON_CV At voice communication */ /* ELIB_REC_ROKUON_TV At TV communication */ } _Elib_REC_STATUS ; *1: For a voice memo waiting for reception, specify at voice communication (ELIB_REC_ROKUON_CV). (See B-1 Record Start Request.) </pre>
Related message	
Pre-conditions	
Note	
Constraints	
Usage	<pre> unsigned int ap_id ; /* Application ID */ int ret ; /* Function return value */ </pre>

```
_Elib_REC_PARAM rec_param ;    /* Record/playbackparameter */
_Elib_REC_STATUS status ;      /* Record information */

/**** Parameter setting ****/
ap_id = AM_AP_ID_JAM_MAIN ;
rec_param.mode = ELIB_REC_DEN ;    /* Verbal message memo */
rec_param.memo_no = ELIB_REC_TELMSG1 ; /* Verbal message memo 1 */

/* Collecting the data information of verbal message memo */
ret = Elib_REC_DataInfo_Get ( ap_id, &rec_param, &status ) ;
```

7.3 Voice Data Count Information Collection

7-3 Voice Data Count Information Collection			
Category	Record/playback service support function		
Description	Collects the voice data count information.		
Name	Elib_REC_DataMaxNumber_Get		
Prototype	Int Elib_REC_DataMaxNumber_Get (unsigned int ap_id, _Elib_REC_PARAM *rec_param, _Elib_REC_KENSU *rec_kensu) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
rec_param	_Elib_REC_P ARAM *	I	Specify the record mode for which count information is to be collected. typedef struct tag_Elib_REC_PARAM { int path ; /* For this function, the path value is invalid. */ int mode ; /* Record/playback mode */ /* The following values can be set. */ /* ELIB_REC_DEN: Verbal message memo */ /* ELIB_REC_VOICE: Voice memo */ /* ELIB_REC_MSG: Fixed message */ int memo_no ; /* For this function, the mode_no value is invalid. */ } _Elib_REC_PARAM ;
rec_kensu	_Elib_REC_K ENSU *	O	Set the following information. typedef struct tag_Elib_REC_KENSU { int data_max ; /* Maximum recording count */ /* in the specified mode */ int data_cnt ; /* Count currently recorded */ /* or registered */ /* Fixed message count */ } _Elib_REC_KENSU ; * If the mode is fixed message, data_max will be an undefined value (unpredictable). * If the return value is other than normal end (ELIB_REC_OK), the value of this argument will be

			undefined (unpredictable).
Return value	Type	I/O	Description
ret	int	O	Normal : ELIB_REC_OK Abnormal : ELIB_REC_NG Incorrect argument value : ELIB_REC_NG_PARM Recording : ELIB_REC_DISCORD
Include file	srv_rec.h		
Functional description	Collects the maximum count that can be recorded, the count currently recorded, and the fixed message count.		
Related message			
Pre-conditions			
Note			
Constraints			
Usage	<pre> unsigned int ap_id ; /* Application ID */ int ret ; /* Function return value */ _Elib_REC_PARAM rec_param ; /* Record/playback parameter */ _Elib_REC_KENSU rec_kensu ; /* For collecting the count information */ /**** Parameter setting ****/ ap_id = AM_AP_ID_JAM_MAIN ; /* Application ID */ rec_param.mode = ELIB_REC_DEN ; /* Verbal message memo */ /* Collecting the verbal message memo count information */ ret = Elib_REC_DataMaxNumber_Get (ap_id, &rec_param, &rec_kensu) ; </pre>		

7.4 Voice Message Memo Start

7-4 Verbal Message Memo Start Setting			
Category	Record/playback service support function		
Description	Sets verbal message memo start.		
Name	Elib_REC_VoiceMemoStatus_Set		
Prototype	int Elib_REC_VoiceMemoStatus_Set (unsigned int ap_id, _Elib_REC_KIDOINFO *kido_info) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
kido_info	_Elib_REC_KIDOINFO *	I	<p>Sets the verbal message memo on/off status and the time up to start.</p> <pre> Typedef struct tag_Elib_REC_KIDOINFO { int on_off ; /* Verbal message memo on/off setting */ /* ON: ELIB_REC_DENGON_ON */ /* OFF: ELIB_REC_DENGON_OFF */ int start_time ; /* Time up to verbal message memo */ /* start (unit: Seconds) */ /* Between 0 and 120 seconds */ /* can be set. */ } _Elib_REC_KIDOINFO ; * If on_off is OFF, the start_time value will be invalid.</pre>
Return value	Type	I/O	Description
ret	Int	O	<p>Normal : ELIB_REC_OK</p> <p>Abnormal : ELIB_REC_NG</p> <p>Incorrect argument value : ELIB_REC_NG_PARM</p>
Include file	srv_rec.h		
Functional description	Sets the verbal message memo on/off and the start time in seconds.		
Related message			

Pre-conditions	
Note	
Constraints	
Usage	<pre> unsigned int ap_id ; /* Application ID */ int ret ; /* Function return value */ _Elib_REC_KIDOINFO kido_info ; /* Verbal message memo setting information */ /**** Parameter setting ****/ ap_id = AM_AP_ID_JAM_MAIN ; /* Application ID */ kido_info.on_off = ELIB_REC_DENGON_ON ; /* Verbal message memo on */ kido_info.start_time = 3 ; /* Setting time 3 seconds */ /* Time setting (seconds) up to verbal message memo on start */ ret = Elib_REC_VoiceMemoStatus_Set(ap_id, &kido_info) ; </pre>

7.5 Verbal Message Memo Start Access

7-5 Verbal Message Memo Start Access			
Category	Record/playback service support function		
Description	Accesses verbal message memo start.		
Name	Elib_REC_VoiceMemoStatus_Get		
Prototype	int Elib_REC_VoiceMemoStatus_Get (unsigned int ap_id, _Elib_REC_KIDOINFO *kido_info) ;		
Argument	Type	I/O	Description
Ap_id	unsigned int	I	Application ID
Kido_info	_Elib_REC_KIDOINFO *	O	<p>Collects the verbal message memo on/off status and the time up to start.</p> <pre>typedef struct tag_Elib_REC_KIDOINFO { int on_off ; /* Verbal message memo on/off setting */ /* ON : ELIB_REC_DENGON_ON */ /* OFF : ELIB_REC_DENGON_OFF */ int start_time ; /* Time up to verbal message memo */ /* start (unit: Seconds) */ /* Between 0 and 120 seconds */ /* can be set. */ } _Elib_REC_KIDOINFO ;</pre> <p>* If on_off is OFF, the start_time value will be undefined (unpredictable).</p> <p>* If the return value is other than normal end (ELIB_REC_OK), the value of this argument will be undefined (unpredictable).</p>
Return value	Type	I/O	Description
Ret	Int	O	<p>Normal : ELIB_REC_OK</p> <p>Abnormal : ELIB_REC_NG</p> <p>Incorrect argument value : ELIB_REC_NG_PARM</p>
Include file	srv_rec.h		
Functional	Collects the verbal message memo on/off status and the time up to start		

description	(unit: Seconds) if the status is on.
Related message	-
Pre-conditions	-
Note	-
Constraints	-
Usage	<pre> unsigned int ap_id ; /* Application ID */ int ret ; /* Function return value */ _Elib_REC_KIDINFO kido_info ; /* Verbal message memo setting information */ /**** Parameter setting ****/ ap_id = AM_AP_ID_JAM_MAIN ; /* Application ID */ /* Collecting the verbal message memo setting status */ ret = Elib_REC_VoiceMemoStatus_Get(ap_id, &kido_info) ; if(ret == ELIB_REC_OK) { /* Checking the verbal mesage memo on/off status and start time */ } </pre>

7.6 Record/Playback Status Access

7-6 Record/Playback Status Access			
Category	Record/playback service support function		
Description	Accesses the record/playback status.		
Name	Elib_REC_Status_Check		
Prototype	int Elib_REC_Status_Check (unsigned int ap_id, _Elib_REC_PARAM *rec_param);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
rec_param	_Elib_REC_PARAM *	O	<p>For record/playback enabled, this parameter will be an undefined value (unpredictable).</p> <p>For recording or playing back, the record/playback destination of current recording and playing back, mode, and data number will be returned.</p> <p>* The confirmation messages are currently being investigated.</p> <pre> typedef struct tag_Elib_REC_PARAM { int path ; /* Record/playback destination specification */ /* The following values are set. */ /* ELIB_REC_UP : Upward; From the local equipment to the communication partner */ /* ELIB_REC_DOWN : Downward; From the communication partner to the local equipment */ int mode ; /* Record/playback mode */ /* The following values are set. */ /* ELIB_REC_DEN : Verbal message memo */ /* ELIB_REC_VOICE : Voice memo */ /* ELIB_REC_MSG : Fixed message */ /* ELIB_REC_SPEECHMSG : Unused */ /* ELIB_REC_SPEECH_SCHALARM : Unused */ /* ELIB_REC_SPEECH_CLKALARM : Unused */ </pre>

```

        */
/* ELIB_REC_SPEECH_TODOALARM :
    Unused          */
/* ELIB_REC_TALKHOLD : Unused */
/* ELIB_REC_OUTOUHORYU
    : Response on-hold tone          */
int memo_no ;
/* Record/playback data number          */
/* The following values are set.          */
/* ELIB_REC_MEMO1 : Voice memo          */
/* ELIB_REC_TELMSG1 : Verbal message
    memo 1          */
/* ELIB_REC_TELMSG2 : Verbal message
    memo 2          */
/* ELIB_REC_TELMSG3 : Verbal message
    memo 3          */
/* ELIB_REC_TELMSG4 : Verbal message
    memo 4          */
/* ELIB_REC_TELMSG5 : Verbal message
    memo 5          */
/* ELIB_REC_MSG_PLAY1 : Fixed message
    (standard)          */
/* ELIB_REC_MSG_PLAY2 : Fixed message
    (private)          */
/* ELIB_REC_MSG_PLAY3
    : Fixed message
    (English)          */
/* ELIB_REC_MSG_CHECK1 : Confirmation
    fixed message (standard) */
/* ELIB_REC_MSG_CHECK2 : Confirmation
    fixed message (private) */
/* ELIB_REC_MSG_CHECK3 : Confirmation fixed
    message (English) */
/* ELIB_REC_SPEECH1 : Unused          */
/* ELIB_REC_SPEECH2 : Unused          */
/* ELIB_REC_OUTOUHORYUMSG1 :
    Response on-hold tone 1          */
/* ELIB_REC_OUTOUHORYUMSG2 :
    Response on-hold tone 2          */
/* ELIB_REC_OUTOUHORYUCHECK1 :

```

```

Confirmation response on-hold tone 1    */
/* ELIB_REC_OUTOUHORYUCHECK2 :
Confirmation response on-hold tone 2    */
/* ELIB_REC_SPEECH_CHECK1    : Unused    */
/*
ELIB_REC_SPEECH_CHECK2    : Unused    */
} _Elib_REC_PARAM ;

* The following values are set at voice memo record/playback.
path                : ELIB_REC_UP,
                    ELIB_REC_DOWN
mode                : ELIB_REC_VOICE
memo_no             : ELIB_REC_MEMO1

* The following values are set at verbal message memo
record/playback.
path                : ELIB_REC_DOWN
mode                : ELIB_REC_DEN
memo_no             : ELIB_REC_TELMSG1 to 5

* The following values are set at fixed message playback.
path                : ELIB_REC_UP
mode                : ELIB_REC_MSG
memo_no             : ELIB_REC_MSG_PLAY1 to 3,
                    ELIB_REC_SPEECH1 to 2

* The following values are set at confirmation fixed message
playback.
path                : ELIB_REC_DOWN
mode                : ELIB_REC_MSG
memo_no             : ELIB_REC_MSG_CHECK1 to
3,
                    ELIB_REC_SPEECH_CHEC
K1 to 2

* The following values are set at response on-hold tone
playback.
path                : ELIB_REC_UP
mode                : ELIB_REC_OUTOUHORYU
memo_no             : ELIB_REC_OUTOUHORYUM
SG1 to 2,
                    ELIB_REC_SPEECH1 to 2

```

			<p>* The following values are set at confirmation response on-hold tone playback.</p> <p>path : ELIB_REC_DOWN</p> <p>mode : ELIB_REC_OUTOUHORYU</p> <p>memo_no : ELIB_REC_OUTOUHORYUCHECK1 to 2, ELIB_REC_SPEECH_CHECK1 to 2</p> <p>* The following values are set at confirmation conversation on-hold tone playback.</p> <p>path : ELIB_REC_DOWN</p> <p>mode : ELIB_REC_TALKHOLD</p> <p>memo_no : ELIB_REC_SPEECH_CHECK1 to 2</p>
Return value	Type	I/O	Description
ret	int	O	<p>Recod and repla possible : ELIB_REC_OK</p> <p>Recording : ELIB_REC_RECORDING</p> <p>playbacking : ELIB_REC_PLAYING</p> <p>Abnormal : ELIB_REC_NG</p> <p>Incorrect argument value : ELIB_REC_NG_PARM</p> <p>Record data being deleted : ELIB_REC_DISCORD</p>
Include file	srv_rec.h		
Functional description	<ul style="list-style-type: none"> - Collects the record/playback status. - If this function is used while record data is being deleted, ELIB_REC_DISCORD will be returned as the return value. 		
Related message			
Pre-conditions			
Note			
Constraints			
Usage	<pre> Unsigned int ap_id ; /* Application */ int ret ; /* Function return value */ _Elib_REC_PARAM rec_param ; /* Record/playback parameter */ /**** Parameter setting ****/ ap_id = AM_AP_ID_JAM_MAIN ; /* Application ID */ </pre>		

```
ret = Elib_REC_Status_Check (ap_id, &rec_param);
                                /* Record/playback status check */

if( ret == ELIB_REC_OK )
{
    rec_param.path = ELIB_REC_UP; /* Upward recording */
    rec_param.mode = ELIB_REC_VOICE; /* Voice memo */
    rec_param.memo_no = ELIB_REC_MEMO1;
                                /* Voice memo 1 */

    /*Starting recording of own voice to voice memo*/
    ret = Elib_REC_Record_Start ( ap_id, &rec_param );
}
```

7.7 Next Playback Data Information Collection

7-7 Next Playback Data Information Collection			
Category	Record/playback service support function		
Description	Collects the next playback data information		
Name	Elib_REC_NextData_Get		
Prototype	int Elib_REC_NextData_Get (unsigned int ap_id, int *memo_no);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
memo_no	int *	O	Number of memo to be played back next Voice memo : ELIB_REC_MEMO1 Verbal message memo 1 : ELIB_REC_TELMSG1 Verbal message memo 2 : ELIB_REC_TELMSG2 Verbal message memo 3 : ELIB_REC_TELMSG3 Verbal message memo 4 : ELIB_REC_TELMSG4 Verbal message memo 5 : ELIB_REC_TELMSG5 * If the return value is other than normal end (ELIB_REC_OK), the value of this argument will be undefined (unpredictable).
Return value	Type	I/O	Description
ret	int	O	Normal end : ELIB_REC_OK Abnormal end : ELIB_REC_NG Incorrect argument value : ELIB_REC_NG_PARM Information read failure : ELIB_REC_FAIL Recording or deleting : ELIB_REC_DISCORD No record data : ELIB_REC_EMPTY
Include file	srv_rec.h		
Functional description	<ul style="list-style-type: none"> - Collects the number of the memo to be played back next. - If this function is called while record data is being played back, the next data number of the record data currently set will be returned as the return value from the record data order storage area. - If this function is called while not recording or playing back, the newest record data number will be returned as the return value from the record data order storage area. - The order of priority of the record data that can be collected is as follows. Verbal message memos (new verbal message memos have priority) > voice 		

	memos
Related message	
Pre-conditions	
Note	<ul style="list-style-type: none"> - If this function is used while record data is being deleted, ELIB_REC_DISCORD will be returned as the return value.
Constraints	
Usage	<p>Assumed conditions :</p> <ol style="list-style-type: none"> 1. The following record dat is present. Voice memo, verbal message memo 1, verbal message memo 2, and verbal message memo 3 2. The order of recording is as follows (the newest is on the left). Verbal message memo 2, verbal message memo 1, voice memo, and verbal message memo 3 <ul style="list-style-type: none"> - Calling this function when not started Return value → Verbal message memo 2 (ELIB_REC_TELMSG2) - Calling this function during playback of verbal message memo 2 Return value → Verbal message memo 1 (ELIB_REC_TELMSG1) - Calling this function during playback of verbal message memo 1 Return value → Verbal message memo 3 (ELIB_REC_TELMSG3) - Calling this function during playback of verbal message memo 3 Return value → Voice memo (ELIB_REC_MEMO1) - Calling this function during playback of voice memo Return value → Verbal message memo 2 (ELIB_REC_TELMSG2) <pre> unsigned int ap_id ; /* Application ID */ int ret ; /* Function return value */ int memo_no ; /* Number of the memo played back */ /** Parameter setting */ ap_id = AM_AP_ID_JAM_MAIN ; /* Application ID */ /* Collecting the verbal message memo setting status */ ret = Elib_REC_NextData_Get (ap_id, &memo_no) ; if(ret == ELIB_REC_OK) { /* Check the memo_no. */ } </pre>

7.8 Verval Message Memo Path Setting

7-8 Verbal Message Memo Path Setting			
Category	Record/playback service support function		
Description	Sets the verbal message memo path.		
Name	Elib_REC_PathSet		
Prototype	int Elib_REC_PathSet (unsigned int ap_id, unsigned char rec_path_mode) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
rec_path_mode	unsigned char	I	Path setting data No connection : ELIB_REC_PATH_OFF Send path connection : ELIB_REC_PATH_TRANS Receive path connection : ELIB_REC_PATH_RECV Send and receive path connection : ELIB_REC_PATH_TALK
Return value	Type	I/O	Description
ret	int	O	Normal : ELIB_REC_OK Abnormal : ELIB_REC_NG Incorrect argument value : ELIB_REC_NG_PARM
Include file	srv_rec.h		
Functional description	<p>- This function is used to set the voice path when a record/playback stop request is generated by the application that is playing back a fixed message or recording a verbal message memo.</p> <p>[Use method] Immediately after a record/playback stop request (Elib_REC_Stop) from an application that is playing back a fixed message or recording a verbal message memo, the following values must be set and this function called.</p> <p>[Only when communication is ended (The end talk key is pressed or disconnection on the network side.)] The second argument must be set to ELIB_REC_PATH_OFF when calling this function.</p> <p>[Only when transiting to a conversation (The start key is pressed.)] The second argument must be set to ELIB_REC_PATH_TALK when calling this function.</p>		
Related	-		

message	
Pre-conditions	-
Note	-
Constraints	-
Usage	<p>* The use method described in the Functional description must be used.</p> <pre> unsigned int ap_id ; /* Application ID */ int ret ; /* Function return value */ unsigned char rec_path_mode ; /* Path setting data */ /** Parameter setting **/ ap_id = AM_AP_ID_JAM_MAIN ; /* Application ID */ rec_path_mode = (unsigned char)ELIB_REC_PATH_OFF ; /* Verbal message memo path setting */ ret = Elib_REC_PathSet(ap_id, rec_path_mode) ; </pre>

7.9 Verval Message Memo Message Pattern Setting

7-9 Verbal Message Memo Message Pattern Setting			
Category	Record/playback service support function		
Description	Sets the verbal message memo message pattern.		
Name	Elib_REC_VoiceMemoPatternSet		
Prototype	int Elib_REC_VoiceMemoPatternSet (unsigned int ap_id, int dmsg_pattern) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
dmsg_pattern	int	I	Verbal message memo message pattern The following values can be set. Standard : ELIB_REC_TELMSGNORM Private : ELIB_REC_TELMSGPRIV English : ELIB_REC_TELMSGENG
Return value	Type	I/O	Description
Ret	int	O	Normal : ELIB_REC_OK Abnormal : ELIB_REC_NG Incorrect argument value : ELIB_REC_NG_PARM
Include file	srv_rec.h		
Functional description	- Sets the verbal message memo message pattern (standard, private, or English).		
Related message			
Pre-conditions			
Note	- When a data exchange service function is used to set or access a response message of a telephone directory specified handy function or group specified handy function, the definition value of the verbal message memo message pattern specified using this function must be used.		
Constraints			
Usage	<pre> unsigned int ap_id ; /* Application ID */ int ret ; /* Function return value */ int dmsg_pattern ; /* For the verbal message memo message pattern */ /** Parameter setting */ ap_id = AM_AP_ID_JAM_MAIN ; /* Application ID */ dmsg_pattern = ELIB_REC_TELMSGNORM ; </pre>		

```
/* Verbal message memo messag pattern standard */
```

```
/* Setting the verbal message memo message pattern */
```

```
ret = Elib_REC_VoiceMemoPatternSet ( ap_id, dmsg_pattern ) ;
```

7.10 Verval Message Memo Message Access

7-10 Verbal Message Memo Message Pattern Access			
Category	Record/playback service support function		
Description	Accesses the verbal message memo message pattern.		
Name	Elib_REC_VoiceMemoPatternGet		
Prototype	int Elib_REC_VoiceMemoPatternGet (unsigned int ap_id, int *dmsg_pattern) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
dmsg_pattern	int *	O	Verbal message memo message pattern The following values are set. Standard : ELIB_REC_TELMSGNORM Private : ELIB_REC_TELMSGPRIV English : ELIB_REC_TELMSGENG * If the return value is other than normal end (ELIB_REC_OK), the value of this argument will be undefined (unpredictable).
Return value	Type	I/O	Description
ret	int	O	Normal : ELIB_REC_OK Abnormal : ELIB_REC_NG Incorrect argument value : ELIB_REC_NG_PARM
Include file	srv_rec.h		
Functional description	- Accesses the verbal message memo memo message pattern (standard, private, or English) that has been set.		
Related message			
Pre-conditions			
Note	- When a data exchange service function is used to set or access a response message of a telephone directory specified handy function or group specified handy function, the definition value of the verbal message memo message pattern specified using this function must be used.		
Constraints			
Usage	<pre> unsigned int ap_id ; /* Application ID */ int ret ; /* Function return value */ int dmsg_pattern ; /* For the verbal message memo message pattern */ </pre>		

```
pattern    */

/** Parameter setting */
ap_id = AM_AP_ID_JAM_MAIN ;    /* Application ID */

/* Accessing the verbal message memo message pattern */
ret = Elib_REC_VoiceMemoPatternGet( ap_id, &dmsg_pattern ) ;
if( ret == ELIB_REC_OK )
{
    /* Check the dmsg_pattern. */
}
```

7.11 Message Title (Response On-Hold Tone and Conversation On-Hand Tone)

7-11 Message Title List (Response On-Hold Tone and Conversation On-Hold Tone)			
Category	Record/playback service support function		
Description	Message title list (response on-hold tone and conversation on-hold tone)		
Name	Elib_REC_Get_Information		
Prototype	<pre>int Elib_REC_Get_Information (unsigned int ap_id, _Elib_REC_GETTTITLEDATA *title_data, _Elib_REC_GETTTITLELIST *title_list) ;</pre>		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
title_data	_Elib_REC_GETTTITLEDATA *	I	<p>Specifies the pointer to the structure used to collect the record/playback data title information.</p> <p>Specifies the number of the record/playback data to be collected.</p> <p>For details, see the Functional description.</p>
title_list	_Elib_REC_GETTTITLELIST *	O	<p>Specifies the pointer to the record/playback data title list structure array.</p> <p>Stores the title data of the record/playback data.</p> <p>For details, see the Functional description.</p> <p>* If the return value is other than normal end (ELIB_REC_NG or ELIB_REC_NG_PARM), the value of this argument will be undefined (unpredictable).</p>
Return value	Type	I/O	Description

ret	int	O	<p>Normal : Collected record/playback data number or the total record/playback data numbers (See the Functional description.)</p> <p>Abnormal : ELIB_REC_NG</p> <p>Incorrect argument value : ELIB_REC_NG_PARM</p>
Include file	srv_rec.h		
Functional description	<p>- The titles of the specified voice data and the record/playback data numbers are provided to the high-level application.</p> <p>- The following functions are supported.</p> <ol style="list-style-type: none"> Response on-hold tone <ul style="list-style-type: none"> (01) ELIB_REC_OUTOUHORYUMSG1 : Response on-hold tone 1 (02) ELIB_REC_OUTOUHORYUMSG2 : Response on-hold tone 2 Conversation on-hold tone <ul style="list-style-type: none"> (01) ELIB_REC_TALKHOLDMSG1 : Conversation on-hold tone <p>Note: The voice memo and verbal message memo titles must be retained on the application side.</p> <p>Second argument <code>_Elib_REC_GETTTITLEDATA</code> Description of the structure for collecting the record/playback data number information</p> <pre>typedef struct tag_Elib_REC_GETTTITLEDATA { int mode; /* Record/playback mode for collecting the titles */ /* ELIB_REC_OUTOUHORYU : Response on-hold tone */ /* ELIB_REC_TALKHOLD : Conversation on-hold tone */ int get_startnum; /* Collection start number of the voice data titles */ /* Set 1 to 4. * See the setting example. */ int get_num; /* Title count of the voice data to be collected */ /* Set 0 to 4. * See the setting example. */ } _Elib_REC_GETTTITLEDATA;</pre> <p>Detailed description of the <code>_Elib_REC_GETTTITLEDATA</code> member mode</p> <p>Specify the mode of the record/playback data for which titles are to be collected.</p> <pre>ELIB_REC_OUTOUHORYU /* Response on-hold tone */ ELIB_REC_TALKHOLD /* Conversation on-hold tone */</pre> <p>Detailed description of the <code>_Elib_REC_GETTTITLEDATA</code> member <code>get_startnum</code></p> <p>Specify the collection start number of the record/playback data titles.</p>		

Data items up to the number specified in `get_num` are collected starting from `get_startnum`.

(The effective value is between 1 and the title total count.)

Detailed description of the `_Elib_REC_GETTTITLEDATA` member `get_num`

Number of record/playback data titles to be collected

When 0 is specified, the titles are not collected. The total number of titles of the specified record/playback

mode is returned as the return value.

Third argument `_Elib_REC_GETTTITLELIST` Description of the record/playback data title list structure

```
typedef struct tag_Elib_REC_GETTTITLELIST{
    int memo_no ;      /* Record/playback data number      */
    int title_len ;    /* Record/playback data character string length */
                        /* The title_len is currently fixed to 0.      */
                        /* 01 indicates that the title character string cannot be edited. */
    char title[ELIB_REC_TITLEMAX+1]
                        /* Record/playback voice data title character string */
                        /* Maximum character string length: ELIB_REC_TITLEMAX = 20 */
} _Elib_REC_GETTTITLELIST ;
```

The relationship between the response on-hold and conversation on-hold data numbers and the voices to be played back is as follows.

`ELIB_REC_OUTOUHORYUMSG1` (response on-hold tone 1)

I cannot answer the phone right now. Please wait or dial again after a little while.

`ELIB_REC_OUTOUHORYUMSG2` (response on-hold tone 2)

I cannot answer the phone right now. Please dial again after a little while.

`ELIB_REC_TALKHOLDMSG1`

Default on-hold tone

<Setting example>

The relationship between the setting and collection data is as follows.

```
title_data->mode          = ELIB_REC_OUTOUHORYU ;      /* Mode      */
title_data->get_startnum   = 2 ;                      /* Second     */
title_data->get_num        = 3 ;                      /* 3          */

title_list[0]->memo_no    == ELIB_REC_OUTOUHORYUMSG2 ;
                        /* First one from the second */
title_list[1]->memo_no    == ELIB_REC_SPEECH1 ;
                        /* Second one */
title_list[2]->memo_no    == ELIB_REC_SPEECH2 ;
                        /* Third one */
```

	<pre> title_list[0]->title[] == "Response on-hold tone 2" ; /* <Note> The title name is access. */ </pre>
Related message	
Pre-conditions	
Note	
Constraints	
Usage	<pre> unsigned int ap_id ; /* Application ID */ int ret ; /* Function return value */ _Elib_REC_GETTTITLEDATA title_data ; /* For collecting the data number information */ _Elib_REC_GETTTITLELIST title_list[3] ; /* Storage area for four data numbers */ /**** Parameter setting ****/ title_data.mode = ELIB_REC_OUTOUHORYU ; title_data.get_startnum = 0 ; ap_id = AM_AP_ID_JAM_MAIN ; /* Application ID */ title_data.get_num = 0 ; /* Collects the total number of data numbers. */ ret = Elib_REC_Get_Informatin (ap_id, &title_data, &title_list[0]) ; /* Accessing the return value ret enables the total number of pages on the title list window to be calculated.*/ /*Next collects the record/playback data numbers of the first and third ones in the record/playback data number list.*/ title_data.mode = ELIB_REC_TALKHOLD /* Collection mode */ title_data.get_startnum = 1 ; /* Starts collecting from the first record/playback data number. */ title_data.get_num = 3 ; /* Collects record/playback data numbers for three titles. */ ret = Elib_REC_Get_Informatin (ap_id, &title_data, &title_list[0]) ; </pre>

7.12 Response On Hold Tone Setting

7-12 Response On-Hold Tone Setting			
Category	Record/playback service support function		
Description	Sets the response on-hold tone.		
Name	Elib_REC_Select_HoldTone		
Prototype	int Elib_REC_Select_HoldTone (unsigned int ap_id, int memo_no) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
memo_no	Int	I	Sets the record/playback data number. The following values can be set. ELIB_REC_OUTOUHORYUMSG1 /* Response on-hold tone 1 */ ELIB_REC_OUTOUHORYUMSG2 /* Response on-hold tone 2 */
Return value	Type	I/O	Description
ret	Int	O	Normal : ELIB_REC_OK Abnormal : ELIB_REC_NG Incorrect argument value : ELIB_REC_NG_PARM
Include file	srv_rec.h		
Functional description	<ul style="list-style-type: none"> - Sets the response on-hold tone. - Use the record/playback data number to specify the response on-hold tone. - The record/playback data number collected using mode = ELIB_REC_OUTOUHORYU for Elib_REC_Get_Information can be used to set the record/playback data number. <p>The relationship between the response on-hold data numbers and the voices to be played back is as follows.</p> <p>ELIB_REC_OUTOUHORYUMSG1 (response on-hold tone 1)</p> <p>I cannot answer the phone right now. Please wait or dial again after a little while.</p> <p>ELIB_REC_OUTOUHORYUMSG2 (response on-hold tone 2)</p> <p>I cannot answer the phone right now. Please dial again after a little while.</p>		
Related			

message	
Pre-conditions	
Note	
Constraints	
Usage	<pre> unsigned int ap_id ; /* Application ID */ int memo_no ; /* Record/playback data number */ int ret ; /* Function return value */ /**** Parameter setting ****/ ap_id = AM_AP_ID_JAM_MAIN ; /* Application ID */ memo_no = ELIB_REC_OUTOUHORYMSG1 ; /* Response on-hold tone setting */ ret = Elib_REC_Select_HoldTone(ap_id, memo_no) ; </pre>

7.13 Response On Hold Tone Access

7-13 Response On-Hold Tone Access			
Category	Record/playback service support function		
Description	Accesses the response on-hold tone.		
Name	Elib_REC_Get_HoldToneTitle		
Prototype	int Elib_REC_Get_HoldToneTitle(unsigned int ap_id, _Elib_REC_GETTITLELIST *title_list);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
title_list	_Elib_REC_GETTITLELIST *	O	Structure of the record/playback data title list * If the return value is other than normal end (ELIB_REC_NG or ELIB_REC_NG_PARM), the value of this argument will be undefined (unpredictable).
Return value	Type	I/O	Description
ret	Int	O	Normal : ELIB_REC_OK Abnormal : ELIB_REC_NG Incorrect argument value : ELIB_REC_NG_PARM
Include file	srv_rec.h		
Functional description	<p>- Accesses the setting information of the response on-hold tone. - Collects the record/playback data numbers and title character string.</p> <p>Description of the record/playback data title list structure</p> <pre> _Elib_REC_GETTITLELIST{ int memo_no /* Record/playback number */ int title_len ; /* Length of title character string of the record/playback data */ char title[ELIB_REC_TITLEMAX+1] ; /* Record/playback voice data title character string */ /* Maximum character string length : ELIB_REC_TITLEMAX = 20 */ }; </pre> <p>Details of the member memo_no of _Elib_REC_GETTITLELIST are as follows.</p> <pre> ELIB_REC_OUTOUHORYUMSG1 /* Response on-hold tone 1 */ ELIB_REC_OUTOUHORYUMSG2 /* Response on-hold tone 2 */ </pre>		

	<p>The relationship between the response on-hold data numbers and the voices to be played back is as follows.</p> <p>ELIB_REC_OUTOUHORYUMSG1 (Response on-hold tone 1)</p> <p>I cannot answer the phone right now. Please wait or dial again after a little while.</p> <p>ELIB_REC_OUTOUHORYUMSG2 (Response on-hold tone 2)</p> <p>I cannot answer the phone right now. Please dial again after a little while.</p>
Related message	
Pre-conditions	
Note	
Constraints	
Usage	<pre> unsigned int ap_id ; /* Application ID */ _Elib_REC_GETTITLELIST title_list ; /* Record/playback data number */ int ret ; /* Function return value */ /** Parameter setting */ ap_id = AM_AP_ID_JAM_MAIN ; /* Application ID */ /* Response on-hold tone access */ ret = Elib_REC_Get_HoldToneTitle (ap_id, &title_list); </pre>

7.14 Downward playback Check

7-14 Downward Playback Check			
Category	Record/playback service support function		
Description	Checks downward playback.		
Name	Elib_REC_Standby_Check		
Prototype	int Elib_REC_Standby_Check(unsigned int ap_id);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Return value	Type	I/O	Description
ret	Int	O	Playback enabled : ELIB_REC_OK Playback disabled : ELIB_REC_NG Incorrect argument value : ELIB_REC_NG_PARM
Include file	srv_rec.h		
Functional description	- Checks downward playback. The result of the downward playback check is set as the return value, as follows. ELIB_REC_OK: playback enabled ELIB_REC_NG: playback disabled		
Related message			
Pre-conditions			
Note			
Constraints			
Usage	<pre> unsigned int ap_id ; /* Application ID */ int ret ; /* Function return value */ /** Parameter setting */ ap_id = AM_AP_ID_JAM_MAIN ; /* Application ID */ /* Downward playback check */ ret = Elib_REC_Standby_Check(ap_id); </pre>		

7.15 TV Verbal Message Memo Response Message Status Registration

7-15 TV Verbal Message Memo Response Message Status Registration			
Category	Record/playback service support function		
Description	Registers the TV verbal message memo response message status.		
Name	Elib_REC_TVRec_Write		
Prototype	int Elib_REC_TVRec_Write(unsigned int ap_id , int mode) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
mode	int	I	TV verbal message memo response message status OFF : ELIB_REC_TVOFF playbacking : ELIB_REC_TVPLAY Recording : ELIB_REC_TVREC
Return value	Type	I/O	Description
ret	Int	O	Normal : ELIB_REC_OK Incorrect argument value : ELIB_REC_NG_PARM
Include file	srv_rec.h		
Functional description	<ul style="list-style-type: none"> Registers the TV verbal message memo response message status. The message status indicates stopping, playbacking, or recording.		
Related message			
Pre-conditions			
Note			
Constraints			
Usage	<pre> unsigned int ap_id ; /* Application ID */ int ret ; /* Function return value */ int mode ; /* TV verbal message memo response message status */ /** Parameter setting **/ ap_id = AM_AP_ID_JAM_MAIN ; mode = ELIB_REC_TVPLAY ; ret = Elib_REC_TVRec_Write(ap_id , mode); </pre>		

7.16 TV Verbal Message Memo Response Message Status Access

7-16 Videophone Verbal Message Memo Response Message Status Access			
Category	Record/playback service support function		
Description	Accesses the TV verbal message memo response message status.		
Name	Elib_REC_TVRec_Read		
Prototype	int Elib_REC_TVRec_Read(unsigned int ap_id);		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Return value	Type	I/O	Description
Ret	Int	O	OFF : ELIB_REC_TVOFF playbacking : ELIB_REC_TVPLAY Recording : ELIB_REC_TVREC Incorrect argument value : ELIB_REC_NG_PARM
Include file	srv_rec.h		
Functional description	- Accesses the TV verbal message memo response message status. The message status indicates stopping, playbacking, or recording.		
Related message			
Pre-conditions			
Note			
Constraints			
Usage	<pre> unsigned int ap_id ; /* Application ID */ int ret ; /* Function return value */ /** Parameter setting **/ ap_id = AM_AP_ID_JAM_MAIN ; /* Application ID */ ret = Elib_REC_TVRec_Read(ap_id); </pre>		

7.17 Conversation On-Hold Tone Setting

7-17 Conversation On-Hold Tone Setting			
Category	Record/playback service support function		
Description	Sets the conversation on-hold tone.		
Name	Elib_REC_Select_Hold		
Prototype	int Elib_REC_Select_Hold(unsigned int ap_id ,int memo_no) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
memo_no	Int	I	<p>Sets the record/playback data number.</p> <p>The following values can be set.</p> <p>ELIB_REC_TALKHOLDMSG1 /* Conversation on-hold tone */</p>
Return value	Type	I/O	Description
Ret	int	O	<p>Normal : ELIB_REC_OK</p> <p>Abnormal : ELIB_REC_NG</p> <p>Incorrect argument value : ELIB_REC_NG_PARM</p>
Include file	srv_rec.h		
Functional description	<p>Sets the conversation on-hold tone.</p> <p>Use the record/playback data number to specify the conversation on-hold tone.</p> <p>The record/playback data number collected using mode = ELIB_REC_TALKHOLD for Elib_REC_GetInformation can be used to set the record/playback data number.</p> <p>The relationship between the conversation on-hold tone data numbers and the voices to be played back is as follows.</p> <p>ELIB_REC_TALKHOLDMSG1</p> <p>Default on-hold tone</p>		
Related message			
Pre-conditions			
Note			
Constraints			
Usage	<p>unsigned int ap_id ; /* Application ID */</p> <p>int memo_no ; /* Playback data number */</p>		

	<pre>int ret ; /* Function return value */ /*** Parameter setting ***/ ap_id = AM_AP_ID_JAM_MAIN ; /* Application ID */ memo_no = ELIB_REC_TALKHOLDMSG1 ; /* Set the record/playback data number. */ ret = Elib_REC_Select_Hold(ap_id , memo_no);</pre>
--	--

7.18 Conversation On-Hold Tone Access

7-18 Conversation On-Hold Tone Access			
Category	Record/playback service support function		
Description	Accesses the conversation on-hold tone.		
Name	Elib_REC_Get_HoldTitle		
Prototype	int Elib_REC_Get_HoldTitle (unsigned int ap_id ,_Elib_REC_GETTITLELIST *title_list) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
title_lsit	_Elib_REC_GETTITLELIST	O	Specifies the pointer to the record/playback data title list structure. Stores the record/playback data number set in nonvolatile memory and the title data. For details, see the Functional description. * If the return value is other than normal end, the value of this argument will be undefined (unpredictable).
Return value	Type	I/O	Description
Ret	Int	O	Normal : ELIB_REC_OK Abnormal : ELIB_REC_NG Incorrect argument value : ELIB_REC_NG_PARM
Include file	srv_rec.h		
Functional description	<ul style="list-style-type: none"> - Accesses the setting information of the conversation on-hold tone. - The record/playback data number and title character string can be collected. Second argument _Elib_REC_GETTITLELIST Description of the record/playback data title list structure <pre>typedef struct tag_Elib_REC_GETTITLELIST { int memo_no; /* Record/playback data number */ int title_len; /* Length of title character string of the record/playback data */ char title[ELIB_REC_TITLEMAX+1]; /* Title character string of the record/playback voice data */ /* (Maximum character string length: ELIB_REC_TITELMAX = 20) */ } _Elib_REC_GETTITLELIST;</pre>		

	<p>Detailed description of the _Elib_REC_GETTITLELIST member memo_no</p> <p>Currently, the number of the record/playback data saved in nonvolatile memory is stored.</p> <p>The following value is set.</p> <pre>ELIB_REC_TALKHOLDMSG1 /* Conversation on-hold tone */</pre> <p>The relationship between the conversation on-hold data numbers and the voices to be played back is as follows.</p> <pre>ELIB_REC_TALKHOLDMSG1</pre> <p>Default on-hold tone</p>
Related message	
Pre-conditions	
Note	
Constraints	
Usage	<pre>unsigned int ap_id ; /* Application ID */ _Elib_REC_GETTITLELIST title_list ; /* Record/playback data title list structure */ int ret ; /* Function return value */ ap_id = AM_AP_ID_JAM_MAIN ; /* Application ID */ ret = Elib_REC_Get_HoldTitle(ap_id , &title_list);</pre>

7.19 Record Start Request

7-19 Record Start Request			
Category	Record/playback service support function		
Description	Requests record start.		
Name	Elib_REC_Record_Start		
Prototype	<pre>int Elib_REC_Record_Start (unsigned int ap_id, _Elib_REC_PARAM *rec_param , _Elib_REC_SUBPARAM *rec_subparam) ;</pre>		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
rec_param	_Elib_REC_PARAM *	I	<p>Specifies the recording destination, recording mode, and record data number.</p> <pre>typedef struct tag_Elib_REC_PARAM { int path ; /* Record/playback destination specification */ /* The following values can be set. */ /* ELIB_REC_UP : Upward; From the local equipment to the conversation partner */ /* ELIB_REC_DOWN ; Downward; From the communication partner to the local equipment */ int mode ; /* Record/playback mode */ /* The following values can be set. */ /* ELIB_REC_DEN : Verbal message memo */ /* ELIB_REC_VOICE : Voice memo */ int memo_no ; /* Record/playback data number */ /* The following values can be set. */ /* ELIB_REC_MEMO1 : Voice memo */ /* ELIB_REC_TELMSG1 : Verbal message memo 1 */ /* ELIB_REC_TELMSG2 : Verbal message memo 2 */ /* ELIB_REC_TELMSG3 : Verbal message memo 3 */ /* ELIB_REC_TELMSG4 : Verbal message memo 4 */</pre>

			<pre> /* ELIB_REC_TELMSG5 : Verbal message memo 5 */ } _Elib_REC_PARAM ; * When ELIB_REC_DEN is specified for the mode, only the receiving voice can be specified for the verbal message memo. Therefore, only ELIB_REC_DOWN can be specifid for the path. If ELIB_REC_UP is specified, ELIB_REC_NG_PARM will be returned. * When ELIB_REC_UP is specified for the path, the voice memo during wait receiving is recorded. In this case, if the mode is not ELIB_REC_VOICE, ELIB_REC_NG_PARM will be returned. (For recording other than that described above, ELIB_REC_DOWN must be set.) </pre>
rec_subparam	_Elib_REC_SUBPARAM *	I	See the Functional description for details about the _Elib_REC_SUBPARAM structure. (The parameters of this structure are not checked within the service.)
Return value	Type	I/O	Description
Ret	Int	O	<p>Normal : The recorded data number is returned.</p> <p>Voice memo : ELIB_REC_MEMO1</p> <p>Verbal message memo 1 : ELIB_REC_TELMSG1</p> <p>Verbal message memo 2 : ELIB_REC_TELMSG2</p> <p>Vrbal message memo 3 : ELIB_REC_TELMSG3</p> <p>Verbal message memo 4 : ELIB_REC_TELMSG4</p> <p>Verbal message memo 5 : ELIB_REC_TELMSG5</p> <p>Abnormal : ELIB_REC_NG</p> <p>Management table full : ELIB_REC_FULL</p> <p>A data number already recorded was specified : ELIB_REC_RECORDED</p> <p>Recording, playbacking, or deleting : ELIB_REC_DISCORD</p> <p>Incorrect argument value : ELIB_REC_NG_PARM</p>
Include file	srv_rec.h		

Functional
description

- Searches the record/playback management table and records the own voice or receiving voice if there is available space in the table.
- Before this function is called, Elib_REC_Request must be used to request notification of the RecordNotify_RPSTOP event. Otherwise, the event will not be posted to the requesting application.
- After recording is stopped, the RecordNotify_RPSTOP event is posted to the requesting application.
- When recording a conventional wait receiving (voice memo), ELIB_REC_UP must be set for the path and ELIB_REC_VOICE for the mode.

- Events that occur when using this function

At recording stop : RecordNotify_RPSTOP

/* Record/playback stop notification */

At recording interrupt : RecordNotify_RECERR

/* Record/playback forced termination notification */

Recording disabled : RecordNotify_RECRXERR

/* Downward voice record start is disabled. */

Recording disabled (wait voice memo only) : RecordNotify_RECTXERR

/* Upward voice record start is disabled. */

* If recording is stopped as described in B-3 Record/playback Stop Request, normal end will be returned.

**** Voice memo, verbal message memo record information
_Elib_REC_SUBPARAM structure ****/

```
typedef struct tag_Elib_REC_SUBPARAM
{
    *1 ee Note.
    Int    month ;    *1
        /* Registration month    (Required for voice and verbal message memos) */
    int    date ;    *1
        /* Registration date      (Required for voice and verbal message memos) */
    int    hour ;    *1
        /* Registration hour      (Required for voice and verbal message memos) */
    int    min ;    *1
        /* Registration minute    (Required for voice and verbal message memos) */
    int    week ;    *1
        /* Registration day of week (Required for voice and verbal message memos) */
        /* Dial buffer (Required for verbal message memos, invalid for voice memos) */
    unsigned char dial_buf [ELIB_REC_DIALMAX] ;
    int    dial_cnt ;    *1
}
```

	<pre> /* Dial counter (Required for verbal message memos, invalid for voice memos) */ /* Character buffer (Required for verbal message memos, invalid for voice memos) */ unsigned char word_buf[ELIB_REC_KANJIMAX]; int word_cnt; *1 /* Character counter (Required for verbal message memos, invalid for voice memos) */ /* En-size read data (Required for verbal message memos, invalid for voice memos) */ unsigned char btyomi [ELIB_REC_KANAMAX]; int btyomilen;*1 /* En-size read data length (Required for verbal message memos, invalid for voice memos) */ int no_info; *1 /* Non-notification incoming information (Required for verbal message memos, invalid for voice memos) */ int grp_num; *1 /* Group number (Required for verbal message memos, invalid for voice memos) */ int icon; /* Multi-icon (Required for verbal message memos, invalid for voice memos) */ int cvtv_flg; /* ELIB_REC_ROKUON_MENU: At menu operations */ /* ELIB_REC_ROKUON_CV: At voice communication */ /* ELIB_REC_ROKUON_TV: At TV communication */ } _Elib_REC_SUBPARAM; * For the structure described above, the values set from the application are used as is. The parameters are not checked within the service. <Note> For voice memos, only the timestamp (month, date, hour, min, and week) is used. For the other members, ELIB_REC_NOTUSE must be used. In addition, if the time is not set, ELIB_REC_NOTDATE must be stored in the timestamp (month, date, hour, min, and week). </pre>
Related message	
Pre-conditions	
Note	<p>Note 1: For * 1 member of the structure _Elib_REC_SUBPARAM, the effective range is within unsigned char. Otherwise, an error will be returned.</p> <ul style="list-style-type: none"> - If an operation error (low voltage) occurs even without a stop request from the application during recording or playbacing, recording or playbacing will be stopped. - If low voltage occurs, record/playback forced termination will be posted.

	<ul style="list-style-type: none"> - Events will not be posted if recording is stopped due to momentary power off. - If this function is used while record data is being deleted, ELIB_REC_DISCORD will be returned as the return value. - During voice memo recording, this function plays the record start notification sound, record stop warning sound, and record stop notification sound only in the following cases. <ul style="list-style-type: none"> - The manner mode and button confirmation sound have been set to off. - The manner mode is set. - During original manner, the memo confirmation sound has been set to on and the button confirmation sound has been set to off in the original manner setting.
Constraints	
Usage	<pre> <Recording a receiving voice> unsigned int ap_id ; /* Application ID */ int ret ; /* Function return value */ _Elib_REC_PARAM rec_param ; /* Record/playback parameter */ _Elib_REC_SUBPARAM rec_subparam ; /* Voice memo verbal message memo record information (sub) */ /** Parameter setting */ ap_id = AM_AP_ID_JAM_MAIN /* Application ID */ rec_param.path = ELIB_REC_DOWN ; /* Downward record */ rec_param.mode = ELIB_REC_DEN ; /* Verbal message memo */ rec_param.memo_no = ELIB_REC_TELMSG1 ; /* Verbal message memo 1 */ rec_subparam.cvtv_flg = ELIB_REC_ROKUON_CV /* At voice communication */ /* Starting recording of the receiving voice to verbal message memo 1 */ ret = Elib_REC_Record_Start (ap_id, &rec_param, &rec_subparam) ; <Recording own voice> unsigned int ap_id ; /* Application ID */ int ret ; /* Function return value */ _Elib_REC_PARAM rec_param ; /* Record/playback parameter */ _Elib_REC_SUBPARAM rec_subparam ; /* Voice memo verbal message memo record information (sub) */ /** Parameter setting */ ap_id = AM_AP_ID_JAM_MAIN ; /* Application ID */ rec_param.path = ELIB_REC_UP ; </pre>

```
/* Upward record */
rec_param.mode = ELIB_REC_VOICE ;
/* Voice memo */
rec_param.memo_no = ELIB_REC_MEMO1 ; /* Voice memo 1 */
rec_subparam.cvtv_flg = ELIB_REC_ROKUON_CV
/* At voice communication */

/* Starting recording of own voice to the available voice memo area */
ret = Elib_REC_Record_Start ( ap_id, &rec_param, &rec_subparam ) ;
```


7.20 Playback Start Request

7-20 Playback Start Request			
Category	Record/playback service support function		
Description	Requests playback start.		
Name	Elib_REC_Play_Start		
Prototype	int Elib_REC_Play_Start (unsigned int ap_id, _Elib_REC_PARAM *rec_param) ;		
Argument	Type	I/O	Description
ap_id	Unsigned int	I	Application ID
rec_param	_Elib_REC_PARAM *	I	<p>Specifies the playback destination, playback mode, and playback data number.</p> <pre> typedef struct tag_Elib_REC_PARAM { int path ; /* Record/playback destination specification */ /* The following values can be set. */ /* ELIB_REC_UP : Upward; From the local equipment to communication partner */ /* ELIB_REC_DOWN : Downward; From the communication partner to local equipment */ int mode ; /* Record/playback mode */ /* The following values can be set. */ /* ELIB_REC_DEN : Verbal message memo */ /* ELIB_REC_VOICE : Voice memo */ /* ELIB_REC_MSG : Fixed message */ /* ELIB_REC_OUTOUHORYU : Response on-hold tone */ /* ELIB_REC_TALKHOLD : Conversation on-hold tone */ int memo_no ; /* Record/playback data number */ /* The following values can be set. */ /* ELIB_REC_MEMO1 : Voice memo */ /* ELIB_REC_TELMSG1 : Verbal message memo 1 */ /* ELIB_REC_TELMSG2 </pre>

			<pre> : Verbal message memo 2 */ /* ELIB_REC_TELMSG3 : Verbal message memo 3 */ /* ELIB_REC_TELMSG4 : Verbal message memo 4 */ /* ELIB_REC_TELMSG5 : Verbal message memo 5 */ /* ELIB_REC_MSG_PLAY1 : Fixed MSG1 (standard) */ /* ELIB_REC_MSG_PLAY2 : Fixed MSG2 (private) */ /* ELIB_REC_MSG_PLAY3 : Fixed MSG3 (English) */ /* ELIB_REC_MSG_CHECK1 : Confirmation fixed MSG1 (standard) */ /* ELIB_REC_MSG_CHECK2 : Confirmation fixed MSG2 (private) */ /* ELIB_REC_MSG_CHECK3 : Confirmation fixed MSG3 (English) */ /* ELIB_REC_OUTOUHORYUMSG1 : Response on-hold tone 1 */ /* ELIB_REC_OUTOUHORYUMSG2 : Response on-hold tone 2 */ /* ELIB_REC_OUTOUHORYUCHECK1 : Confirmation response on-hold tone 1 */ /* ELIB_REC_OUTOUHORYUCHECK2 : Confirmation response on-hold tone 2 */ /* ELIB_REC_SPEECH1 : Unused */ /* ELIB_REC_SPEECH2 : Unused */ /* ELIB_REC_SPEECH_CHECK1 : Unused */ /* ELIB_REC_SPEECH_CHECK2 : Unused */ /* ELIB_REC_TALKHOLDMSG1 : Default on-hold tone */ } _Elib_REC_PARAM ; * When ELIB_REC_UP is specified for the path, ELIB_REC_DEN or ELIB_REC_VOICE cannot be set for the mode. If set, ELIB_REC_NG_PARM will be returned. * For voice memo playback, set the following. </pre>
--	--	--	---

			<p>path : ELIB_REC_DOWN</p> <p>mode : ELIB_REC_VOICE</p> <p>memo_no : ELIB_REC_MEMO1</p> <p>* For verbal message memo playback, set the following.</p> <p>Path : ELIB_REC_DOWN</p> <p>mode : ELIB_REC_DEN</p> <p>memo_no : ELIB_REC_TELMSG1 to 5</p> <p>* For fixed message playback (upward playback), set the following.</p> <p>Path : ELIB_REC_UP</p> <p>Mode : ELIB_REC_MSG</p> <p>memo_no : ELIB_REC_MSG_PLAY1 to 3, ELIB_REC_SPEECH1 to 2</p> <p>* For confirmation fixed message playback (downward playback), set the following.</p> <p>Path : ELIB_REC_DOWN</p> <p>mode : ELIB_REC_MSG</p> <p>memo_no : ELIB_REC_MSG_CHECK1 to 3, ELIB_REC_SPEECH_CHECK1 to 2</p> <p>* For response on-hold tone playback (upward playback), set the following.</p> <p>At response on-hold tone playback (upward playback), playback is repeated until the B-3 Record/playback Stop Request (Elib_REC_STOP) is called.</p> <p>Path : ELIB_REC_UP</p> <p>mode : ELIB_REC_OUTOUHORYU</p> <p>memo_no : ELIB_REC_OUTOUHORYUMSG1 to 2, ELIB_REC_SPEECH1 to 2</p> <p>* For confirmation response on-hold tone playback (downward playback), set the following.</p> <p>path : ELIB_REC_DOWN</p> <p>mode : ELIB_REC_OUTOUHORYU</p> <p>memo_no : ELIB_REC_OUTOUHORYUCHECK 1 to 2,</p>
--	--	--	---

			<p>ELIB_REC_SPEECH_CHECK1 to 2</p> <p>* For confirmation conversation on-hold tone playback (downward playback), set the following.</p> <p>path : ELIB_REC_DOWN</p> <p>mode : ELIB_REC_TALKHOLD</p> <p>memo_no : ELIB_REC_SPEECH_CHECK1 to 2</p>
Return value	Type	I/O	Description
ret	int	O	<p>Normal : The played data number is returned.</p> <p>ELIB_REC_MEMO1 : Voice memo</p> <p>ELIB_REC_TELMSG1 : Verbal message memo 1</p> <p>ELIB_REC_TELMSG2 : Verbal message memo 2</p> <p>ELIB_REC_TELMSG3 : Verbal message memo 3</p> <p>ELIB_REC_TELMSG4 : Verbal message memo 4</p> <p>ELIB_REC_TELMSG5 : Verbal message memo 5</p> <p>ELIB_REC_MSG_PLAY1 : Fixed MSG1</p> <p>ELIB_REC_MSG_PLAY2 : Fixed MSG2</p> <p>ELIB_REC_MSG_PLAY3 : Fixed MSG3</p> <p>ELIB_REC_MSG_CHECK1 : Confirmation fixed MSG1</p> <p>ELIB_REC_MSG_CHECK2 : Confirmation fixed MSG2</p> <p>ELIB_REC_MSG_CHECK3 : Confirmation fixed MSG3</p> <p>ELIB_REC_OUTOUHORYUMSG1 : Response on-hold tone 1</p> <p>ELIB_REC_OUTOUHORYUMSG2 : Response on-hold tone 2</p> <p>ELIB_REC_OUTOUHORYUCHECK1 : Confirmation response on-hold tone 1</p> <p>ELIB_REC_OUTOUHORYUCHECK2 : Confirmation response on-hold tone 2</p> <p>ELIB_REC_SPEECH1 : Unused</p> <p>ELIB_REC_SPEECH2 : Unused</p>

			ELIB_REC_SPEECH_CHECK1 : Unused ELIB_REC_SPEECH_CHECK2 : Unused Abnormal : ELIB_REC_NG The specified record data cannot be found : ELIB_REC_EMPTY Recording, playing, or deleting : ELIB_REC_DISCORD Incorrect argument value : ELIB_REC_NG_PARM
Include file	srv_rec.h		
Functional description	<ul style="list-style-type: none"> - Plays the specified voice data. - Before this function is called, Elib_REC_Request must be used to request notification of the event corresponding to the specified voice data. Otherwise, the event will not be posted to the requesting application. - After playback is stopped, the event corresponding to the specified voice data is posted to the requesting application. - Events that occur when using this function <ul style="list-style-type: none"> At verbal message memo/voice memo playback start notification : RecordNotify_PLAYSTART /* Playback start */ At playback stop stop notification : RecordNotify_RPSTOP /* Record/playback */ At playback interrupt forced termination notification : RecordNotify_RECERR /* Record/playback */ Playback disabled playback start is disabled. : RecordNotify_PLAYRXERR /* Downward voice */ - Fixed message/response on-hold tone <ul style="list-style-type: none"> At playback start (at downward playback only) notification : RecordNotify_PLAYSTART /* Playback start */ At playback normal end completion notification : RecordNotify_RECMSGSTOP /* Fixed message */ At playback interrupt forced termination notification : RecordNotify_RECERR /* Record/playback */ Playback disabled (only at upward playback) playback start is disabled. : RecordNotify_PLAYTXERR /* Upward voice */ Playback disabled (only at downward playback) playback start is disabled. : RecordNotify_PLAYRXERR /* Downward voice */ 		

	<ul style="list-style-type: none"> * Playback interrupt means that processing was interrupted within the handler. If playback is stopped using the B-3 Record/Playback Stop Request, normal end will be returned. * If processing is stopped within the handler at upward playback of a fixed message, RecordNotify_RECMSGSTOP will be posted and ELIB_REC_NG set in the member info of the event structure. For details, see A-1 Record/Playback Event Notification Request Start.
Related message	
Pre-conditions	
Note	<ul style="list-style-type: none"> - If an operation error (low voltage) occurs even without a stop request from the application during recording or playbacks, recording or playbacks will be stopped. - If low voltage occurs, record/playback forced termination will be posted. - Events will not be posted if playbacks is stopped due to momentary power off. - If this function is used while record data is being deleted, ELIB_REC_DISCORD will be returned as the return value and playback will not be performed. - When playbacks voice or verbal message memos, the A-3 Record Data Information Collection Request (ELIB_REC_DataInfo_Get) must be used to check that the record data to be playbacks is present. That is, this function must be used only when record data is present. - When calling this function with ELIB_REC_DOWN specified for the member path of the argument structure, call A-20 Downward Playback Check (Elib_REC_Standby_Check) and check downward playback before calling this function. Do not call this function if the return value is other than playback enabled. - At voice memo or verbal message memo playback, this function plays the playback start notification tone and playback stop notification tone only in the following cases. <ul style="list-style-type: none"> - The manner mode and button confirmation sound have been set to off. - During manner mode - During original manner, the memo confirmation sound has been set to on and the button confirmation sound has been set to off in the original manner setting.
Constraints	
Usage	<pre> [Playbacks a fixed message to the other party during a conversation] unsigned int ap_id ; /* Application ID */ int ret ; /* Function return value */ _Elib_REC_PARAM rec_param ; /* Record/playback parameter */ </pre>

```

    /*** Parameter setting ***/
    ap_id = AM_AP_ID_JAM_MAIN ;                               /* Application
    ID                                                         */
    rec_param.path = ELIB_REC_UP ;                             /* Upward
    playback                                                */
    rec_param.mode = ELIB_REC_MSG ;                             /* Fixed
    message                                                */
    rec_param.memo_no = ELIB_REC_MSG_PLAY1 ;                 /* Fixed message
    1                                                         */

    /* Starting playbacing of fixed message 1 to other party during conversation */
    ret = Elib_REC_Play_Start ( ap_id, &rec_param ) ;

    [Playbacing a voice memo that has not been playbaced during wait receiving]
    unsigned int ap_id ;                                       /* Application
    ID                                                         */
    int ret ;                                                  /* Function
    return value                                              */
    _Elib_REC_PARAM
    rec_param ;                                               /* Record/playback
    parameter                                                */

    /*** Parameter setting ***/
    ap_id = AM_AP_ID_JAM_MAIN ;                               /* Application
    ID                                                         */

    /* Checking that downward playback is enabled */
    ret = Elib_REC_Standby_Check( ap_id ) ;
    if ( ret == ELIB_REC_OK )
    {
        rec_param.path = ELIB_REC_DOWN ;                     /* Downward
        */
        rec_param.mode = ELIB_REC_VOICE ;                     /* Voice
        memo                                                */
        rec_param.memo_no = ELIB_REC_MEMO1 ;                 /* Voice memo
        1                                                         */

        /* Starting playbacing of a voice memo */
        ret = Elib_REC_Play_Start ( ap_id, &rec_param ) ;

```

```

}

[Starting playbacing of a response on-hold message for response on-hold]
unsigned int ap_id ;                               /* Application
ID                                                  */
int ret ;                                          /* Function
return value                                       */
_Elib_REC_PARAM
rec_param ;                                       /* Record/playback
parameter                                         */

/**/ Parameter setting /**/
ap_id =
AM_AP_ID_JAM_MAIN ;                             /* Application
ID                                                  */
rec_param.path = ELIB_REC_UP ;                   /* Upward
playback                                           */
rec_param.mode = ELIB_REC_OUTOUHORYU ;           /* Response
on-hold tone                                       */
rec_param.memo_no = ELIB_REC_OUTOUHORYUMSG1 ;    /* Response on-hold
tone 1                                             */

/* Starting playbacing of response on-hold tone 1 to the other party during a
conversation */
ret = Elib_REC_Play_Start ( ap_id, &rec_param ) ;

[Starting playbacing of a confirmation response on-hold tone selected using the
response on-hold menu]
unsigned int ap_id ;                               /* Application
ID                                                  */
int ret ;                                          /* Function
return value                                       */
_Elib_REC_PARAM
rec_param ;                                       /* Record/playback
parameter                                         */

/**/ Parameter setting /**/
ap_id = AM_AP_ID_JAM_MAIN ;                       /* Application
ID                                                  */

/* Checking that downward playback is enabled */
ret = Elib_REC_Standby_Check( ap_id ) ;
if( ret == ELIB_REC_OK )

```

```
{
    rec_param.path    =    ELIB_REC_DOWN    ;                                /*
Downward                                                    */
    rec_param.mode    =    ELIB_REC_OUTOUHORYU    ;                        /* Response
on-hold tone                                                */
    rec_param.memo_no = ELIB_REC_OUTOUHORYUCHECK 1 ;                      /* Confirmation
response on-hold tone 1  */

    /* Starting playbacking of response on-hold tone 1 on the mobile equipment side */
    ret = Elib_REC_Play_Start ( ap_id,  &rec_param ) ;
}
```

7.21 Record/Playback Stop Request

7-21 Record/Playback Stop Request			
Category	Record/playback service support function		
Description	Requests record/playback stop.		
Name	Elib_REC_Stop		
Prototype	int Elib_REC_Stop (unsigned int ap_id) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
Return value	Type	I/O	Description
Ret	Int	O	Normal : ELIB_REC_OK Abnormal : ELIB_REC_NG Incorrect argument value : ELIB_REC_NG_PARM
Include file	srv_rec.h		
Functional description	<ul style="list-style-type: none"> - Stops record/playback during conversation or wait receiving. - Before this function is called, Elib_REC_Request must be used to request notification of the event corresponding to the voice that was stopped automatically. Otherwise, the event will not be posted to the requesting application. - After record/playback is stopped, the event corresponding to the voice data that was stopped is posted to the requesting application. - Events that occur when using this function <ul style="list-style-type: none"> - Verbal message memo/voice memo Playback stop notification: RecordNotify_RPSTOP /* Record/playback stop notification */ - Fixed message/response on-hold tone Fixed message completion notification: RecordNotify_RECMSGSTOP /* Fixed message completion notification */ 		
Related message			
Pre-conditions			
Note	<ul style="list-style-type: none"> - When this function is used during fixed message playback or verbal message memo recording, the Verbal Message Memo Path Setting Function must be used to set the voice path after this function has been used. (For details see A-9.) 		
Constraints			
Usage	unsigned int ap_id ; /* Application ID */ int ret ; /* Function return value */		

```
/** Parameter setting */
ap_id = AM_AP_ID_JAM_MAIN ;          /* Application ID */

/* Stopping record/playback */
ret = Elib_REC_Stop ( ap_id ) ;
```


7.22 Record Data Deletion Request

7-22 Record Data Deletion Request			
Category	Record/playback service support function		
Description	Requests deletion of record data.		
Name	Elib_REC_Data_Erase		
Prototype	int Elib_REC_Data_Erase (unsigned int ap_id, _Elib_REC_PARAM * rec_param) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
rec_param	_Elib_REC_PARAM *	I	<p>Specifies the record mode and data number to be deleted.</p> <pre> typedef struct tag_Elib_REC_PARAM { int path ; /* For this function, the path value is invalid. */ int mode ; /* Record/playback mode */ /* The following values can be set. */ /* ELIB_REC_DEN: Verbal message memo */ /* ELIB_REC_VOICE: Voice memo */ /* ELIB_REC_ALL: All modes (verbal message memo/voice memo) */ int memo_no ; /* Record/playback data number */ /* The following values can be set. */ /* ELIB_REC_CKUNT All record data of the specified mode is deleted. */ /* ELIB_REC_MEMO1: Voice memo */ /* ELIB_REC_TELMSG1 : Verbal message memo 1 */ /* ELIB_REC_TELMSG2 : Verbal message memo 2 */ /* ELIB_REC_TELMSG3 : Verbal message memo 3 */ /* ELIB_REC_TELMSG4 : Verbal message memo 4 */ /* ELIB_REC_TELMSG5 : Verbal message memo 5 */ } _Elib_REC_PARAM ; </pre> <p>* When ELIB_REC_ALL has been set for the mode, all record data of the verbal message and voice memos is deleted. At this</p>

			<p>time, the memo_no value is invalid.</p> <p>* When other than ELIB_REC_ALL has been set for the mode and ELIB_REC_CKUNT has been set for the memo_no, all of the data of the specified record mode is deleted.</p> <p>* When ELIB_REC_DEN has been set for the mode and ELIB_REC_MEMO1 for the memo_no or ELIB_REC_VOICE has been set for the mode and ELIB_REC_TELMSG1 to 5 for the memo_no, ELIB_REC_NG_PARM will be returned.</p>
Return value	Type	I/O	Description
Ret	int	O	<p>Normal : ELIB_REC_OK</p> <p>Abnormal : ELIB_REC_NG</p> <p>No record data : ELIB_REC_EMPTY</p> <p>Playbacking or recording : ELIB_REC_DISCORD</p> <p>Incorrect argument value : ELIB_REC_NG_PARM</p>
Include file	srv_rec.h		
Functional description	<ul style="list-style-type: none"> - Deletes the specified record data or all of the record data. - Before this function is called, Elib_REC_Request must be used to request notification of the event corresponding to RecordNotify_ERASE. Otherwise, the event will not be posted to the requesting application. - After deletion is completed, the RecordNotify_ERASE event is posted to the requesting application. - At total deletion or verbal message memo total deletion, the event (RecordNotify_ERASE) is posted one time only after total deletion is stopped regardless of the number of data items deleted. - Event that occurs when using this function Voice data deletion completed: RecordNotify_ERASE /* Record data deletion completed notification */ 		
Related message			
Pre-conditions			
Note			
Constraints			
Usage	<p>[Deleting a single verbal message memo]</p> <pre> unsigned int ap_id ; /* Application ID */ int ret ; /* Function return value */ _Elib_REC_PARAM rec_param ; /* Record/playback parameter */ </pre>		

```

/** Parameter setting */
ap_id = AM_AP_ID_JAM_MAIN ;          /* Application ID */
rec_param.mode = ELIB_REC_DEN ;
                                     /* Verbal message memo */
rec_param.memo_no = 1 ;              /* Data number 1 */

/* Record data deletion */
ret = Elib_REC_Data_Erase ( ap_id, &rec_param ) ;
                                     /* Deletes verbal message memo 1. */

[Deleting all verbal message memos]
unsigned int ap_id ;                  /* Application ID */
int ret ;                            /* Function return value */
_Elib_REC_PARAM rec_param ;          /* Record/playback parameter */

/** Parameter setting */
ap_id = AM_AP_ID_JAM_MAIN ;          /* Application ID */
rec_param.mode = ELIB_REC_DEN ;      /* Verbal message memo */
rec_param.memo_no = ELIB_REC_CKCOUNT ;
                                     /* Deletes all record data of the specified mode. */

/* Record data deletion */
ret = Elib_REC_Data_Erase ( ap_id, &rec_param ) ;
                                     /* Deletes all verbal message memos. */

[Deleting all record data]
unsigned int ap_id ;                  /* Application ID */
int ret ;                            /* Function return value */
_Elib_REC_PARAM rec_param ;          /* Record/playback parameter */

/** Parameter setting */
ap_id = AM_AP_ID_JAM_MAIN ;          /* Application ID */
rec_param.mode = ELIB_REC_ALL ;      /* All modes */

/* Record data deletion */
ret = Elib_REC_Data_Erase ( ap_id, &rec_param ) ;
                                     /* Deletes all verbal message memos and voice memos. */

```

7.23 Record Start Request (Confirmation Tone Flag Attached)

7-23 Record Start Request (Confirmation Tone Flag Attached)			
Category	Record/playback service support function		
Description	Requests record start (confirmation tone flag attached).		
Name	Elib_REC_Key_Start		
Prototype	<pre>int Elib_REC_Key_Start (unsigned int ap_id, _Elib_REC_PARAM *rec_param, _Elib_REC_SUBPARAM *rec_subparam, unsigned char sound_on_off) ;</pre>		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
rec_param	_Elib_REC_PARAM *	I	<p>Specifies the record destination, record mode, and record data number.</p> <p>* For details about the contents, see B-1 Record Start Request. (There are no differences from B-1.)</p>
rec_subparam	_Elib_REC_SUBPARAM *	I	<p>* For details about the contents, see B-1 Record Start Request. (There are no differences from B-1.)</p>
sound_on_off	unsigned char	I	<p>Confirmation sound play present/absent</p> <p>For voice memo recording during conversation after the side key has been pressed for a long time, the following is set.</p> <p>ELIB_REC_SS_ON : Confirmation sound play present</p> <p>* Currently, this function can only be used for the above.</p>
Return value	Type	I/O	Description
Ret	Int	O	<p>* For details about the contents, see B-1 Record Start Request. (There are no differences from B-1.)</p>
Include file	srv_rec.h		
Functional	- This function is the same as the B-1 Record Start Request. (For details, see B-1.)		

description	<ul style="list-style-type: none"> - The differences from B-1 are as follows. A fourth argument has been added. * When recording is started without the start sound (only when voice memo record start is requested by pressing the side key for a long time during conversation), ELIB_REC_SS_ON is set in the fourth argument.
Related message	
Pre-conditions	
Note	<ul style="list-style-type: none"> - This function must be used only for voice memo record start requests by pressing the side key for a long time during conversation. (Currently, this function can only be used for the above.) - For record start requests other than the above, the B-1 Record Start Request must be used.
Constraints	
Usage	<ul style="list-style-type: none"> * For a usage, see B-1 Record Start Request. <p>Note: This function must be used only for recording voice memos by pressing the side key for a long time during conversation. For record start requests other than the above, the B-1 Record Start Request must be used.</p>

7.24 Playback Start Request(Confirmation Tone Flag Attached)

7-24 Playback Start Request (Confirmation Tone Flag Attached)			
Category	Record/playback service support function		
Description	Requests playback start (confirmation tone flag attached).		
Name	Elib_REC_SideKeyPlayStart		
Prototype	<pre>int Elib_REC_SideKeyPlayStart (unsigned int ap_id, _Elib_REC_PARAM *rec_param, unsigned char sound_on_off);</pre>		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
rec_param	_Elib_REC_PARAM *	I	<p>Specifies the playback destination, playback mode, and playback data number.</p> <p>* For details about the contents, see B-2 Playback Start Request. (There are no differences from B-2.)</p>
sound_on_off	unsigned char	I	<p>Confirmation tone play present/absent</p> <p>For memo playback requests by pressing the side key for a short time after waiting, the following is set.</p> <p style="text-align: center;">ELIB_REC_SS_ON : Confirmation tone play present</p> <p>* Currently, this function can only be used for the above.</p>
Return value	Type	I/O	Description
Ret	int	O	<p>* For details about the contents, see B-2 Playback Start Request. (There are no differences from B-2.)</p>
Include file	srv_rec.h		
Functional description	<ul style="list-style-type: none"> - This function is the same as the B-2 Playback Start Request. (For details, see B-2.) - The differences from B-2 are as follows. A third argument has been added. * When playback is started without the start sound (only when memo playback start is requested by pressing the side key for a short time during waiting), ELIB_REC_SS_ON is set in the third argument. 		

Related message	
Pre-conditions	
Note	<ul style="list-style-type: none">- This function must be used only for memo playback start requests by pressing the side key for a short time during waiting. (Currently, this function can only be used for the above.)- For playback start requests other than the above, the B-2 Playback Start Request must be used.- This function must be used only for playnbacking memos by pressing the side key for a short time during waiting. For playback requests other than the above, the B-2 Playback Start Request must be used.
Constraints	
Usage	* For a usage, see B-2 Playback Start Request.

7.25 Record/Playback Stop Request (Confirmation Tone Flag Attached)

7-25 Record/Playback Stop Request (Confirmation Tone Flag Attached)			
Category	Record/playback service support function		
Description	Requests record/playback stop (confirmation tone flag attached).		
Name	Elib_REC_Key_Stop		
Prototype	int Elib_REC_Key_Stop (unsigned int ap_id, unsigned char sound_on_off) ;		
Argument	Type	I/O	Description
ap_id	unsigned int	I	Application ID
sound_on_off	unsigned char	I	<p>Confirmation tone play present/absent</p> <p>The following value is set when memo playback stop (stopping of the memo currently being played) is requested before the next memo is played by pressing the side key during memo playback.</p> <p style="text-align: center;">ELIB_REC_SS_OFF: Confirmation tone play absent</p> <p>* Currently, this function can only be used for the above.</p>
Return value	Type	I/O	Description
Ret	int	O	<p>* For details about the contents, see B-3 Record/Playback Stop Request. (There are no differences from B-3.)</p>
Include file	srv_rec.h		
Functional description	<ul style="list-style-type: none"> - This function is the same as the B-3 Record/Playback Stop Request. (For details, see B-3.) - The differences from B-3 are as follows. A second argument has been added. <p>* When stopping of memo playback (stopping the memo currently being played) is requested before the next memo is played by pressing the side key during memo playback, ELIB_REC_SS_OFF is set in the second argument.</p>		
Related message			
Pre-conditions			

Note	<ul style="list-style-type: none">- This function must be used only for playback stop requests by pressing the side key during memo playback. (Currently, this function can only be used for the above.)- For stop requests other than the above, the B-3 Record/Playback Stop Request must be used.- This function must be used only for stopping the memo currently being played back before the next memo is played back by pressing the side key during memo playback.- For stop requests other than the above, the B-3 Record/Playback Stop Request must be used.
Constraints	
Usage	* For a usage, see B-3 Record/Playback Stop Request.

8. LMP MANAGEMENT SERVICE

8.1 Request to Turn Backlight On

8-1 Request to Turn Backlight On			
Classification	Function group provided by the LMP management service shared library		
Function name	Request to turn backlight on		
Symbol	Elib_LMP_Backlight_On		
Format	<pre>int Elib_LMP_Backlight_On (unsigned int ap_id , int reason, _ELIB_LMP_BKLT_CONFIGSET *bklt_cfgset) ;</pre>		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
Reason	Int	I	Reason for turning the backlight on Used to solve competitive operation. (See the function explanation for details.)
bklt_cfgset	_ELIB_LMP_BKLT_CONFIGSET *	I	Backlight lighting attributes Specifies the following lighting attributes. (See the function explanation for details.) (1) Backlight color (2) Backlight lighting mode (3) Backlight lighting time
Return value	Type	I/O	Explanation
Ret	int	O	Normal end: ELIB_LMP_OK Cancelled to prioritize other requests: ELIB_LMP_CANCEL Parameter error: ELIB_LMP_NG
Include file	srv_lmp.h		
Function explanation	<p>This function is used to set the backlight lighting attributes and turn the backlight on.</p> <p>Details of reason (second argument) Specify this argument to determine how the backlight is turned on or off based on the current lighting status and reason for turning it on when the request is sent. Specify one of the following values in reason:</p> <pre>ELIB_LMP_BKLT_RCV /* Normal incoming call */ (When the call on hold is received, send a request to turn the backlight on with on for 15 seconds specified.) ELIB_LMP_BKLT_RCVREJECT /* Incoming call rejected */ ELIB_LMP_BKLT_RCVREC /* Answer phone */ ELIB_LMP_BKLT_TRNSONHOOK /* On-hook outgoing */ ELIB_LMP_BKLT_TRNSOFFHOOK /* Off-hook outgoing */ ELIB_LMP_BKLT_TERMDISCONN/* Button operation (mobile phone call-ending) */</pre>		

ELIB_LMP_BKLT_OTHERDISCONN	/* Disconnected by the remote side */ (Specify this value when the remote side of the incoming call disconnects the call.)
ELIB_LMP_BKLT_MSGAUTO	/* Automatic message display */
ELIB_LMP_BKLT_TRNSTOTALK	/* Communication from outgoing */ (Can be omitted)
ELIB_LMP_BKLT_RCVTOTALK	/* Communication from incoming */
ELIB_LMP_BKLT_RSPWAIT	/* Answer on hold */
ELIB_LMP_BKLT_RECDENGON	/* Message slip */
ELIB_LMP_BKLT_RECVOICE	/* Recording voice slip */
ELIB_LMP_BKLT_RECPLAY	/* Playbacking voice slip */
ELIB_LMP_BKLT_SCHEDULE	/* Schedule alarm */
ELIB_LMP_BKLT_RCVPACKET	/* Incoming packet call */
ELIB_LMP_BKLT_ENDPACKET	/* Incoming packet call completed (See (6) in Notes.) */
ELIB_LMP_BKLT_TERMOPEN	/* From mobile phone closed to open */
ELIB_LMP_BKLT_TERMCLOSE	/* From mobile phone open to closed */
ELIB_LMP_BKLT_POWERON	/* Power on */
ELIB_LMP_BKLT_POWEROFF	/* Power off */
ELIB_LMP_BKLT_IMGETCONTENT	/* Content collection operation */
ELIB_LMP_BKLT_IMENDCONTENT	/* Content collection completed */
ELIB_LMP_BKLT_IMODEMSG	/* Mail or message (display or display continuation)(See (9) in Notes.) */
	/* (Read, submenu display, or detailed display of other messages) */
ELIB_LMP_BKLT_IMODEMSGEXIT	/* Mail or message display completed (shift to other, etc.) */
ELIB_LMP_BKLT_JAVA	/* During JAVA operation (during execution of KVM/JAVA application) */
ELIB_LMP_BKLT_JAM	/* During JAVA operation (JAM) */
ELIB_LMP_BKLT_LOWBATTERY	/* Low voltage alarm */
ELIB_LMP_BKLT_IMOTION	/* At video playback (See (7) in Notes.) */
ELIB_LMP_BKLT_UDRCV	/* At Unrestricted digital data reception */
ELIB_LMP_BKLT_CAMERA	/* After shooting still images with the camera function */
ELIB_LMP_BKLT_ACAMERA	/* At shooting moving images with the camera function (See (8) in Notes.) */
ELIB_LMP_BKLT_TVTALK	/* TV phone communication started (at screen display) * Change from incoming to communicating, or from outgoing to communicating */
ELIB_LMP_BKLT_CAMERA_VIEW	/* Camera view display */
ELIB_LMP_BKLT_ADL_REWRITE	/* ADL software update */
Details of bkl_t_cfgset (third argument)	
/* Structure for setting backlight lighting attributes */	
typedef struct tag_Elib_LMP_BKLT_ConfigSet {	
unsigned short	BKLT_Color; /* Backlight color */
unsigned short	BKLT_Mode; /* Backlight lighting mode */

	<pre> unsigned short BKLT_KEY_Time; /* Backlight lighting time */ } _ELIB_LMP_BKLT_CONFIGSET ; </pre> <p>(1) Backlight color: ELIB_LMP_BKLT_COLOR1 (this is a fixed value.)</p> <p>(2) Backlight lighting mode: ELIB_LMP_BKLT_ALWAYS: Always on ELIB_LMP_BKLT_DURING: Turns off after continuing on for a given period of time</p> <p>(3) Backlight lighting time: When ELIB_LMP_BKLT_DURING is specified in the lighting mode, specify the lighting time in seconds using a value from 1 to 100.</p>
Related message	-
Required procedure	-
Notes	<p>(1) When a request to turn the backlight on is sent for the current backlight, the LMP management service determines how the backlight is turned on based on the current lighting status and reason for turning it on.</p> <p>(2) When a request to turn the backlight on is sent for the current backlight, the LMP management service may determine that the current lighting settings are not updated based on the current lighting status and reason for turning it on. In this case, the service changes nothing (does not update the lighting attributes).</p> <p>(3) When ELIB_LMP_BKLT_ALWAYS (Always on) is specified in the lighting mode, the lighting time is invalid. (The setting is not reflected.)</p> <p>(4) While the device is being closed, the LMP management service does not turn the backlight on. Instead, it returns return value ELIB_LMP_CANCEL.</p> <p>(5) When ELIB_LMP_BKLT_ALWAYS is specified in the display setting of the backlight being recharged, the backlight always comes on during device recharge.</p> <p>(6) When the backlight need not be turned on according to the setting condition of the i-mode ringing time (e.g., packet reception completed screen), applications must take action not to call this function.</p> <p>(7) In video playback operation, ELIB_LMP_BKLT_ALWAYS may be specified while ELIB_LMP_BKLT_IMOTION is specified in the reason for turning the backlight on. In this case, specify ELIB_LMP_BKLT_IMOTION and send a request to turn the backlight on for 15 seconds at stop or interruption of video playback.</p> <p>(8) When shooting moving images using the camera function, ELIB_LMP_BKLT_ALWAYS may be specified while ELIB_LMP_BKLT_ACAMERA is specified in the reason for turning the backlight on. In this case, specify ELIB_LMP_BKLT_ACAMERA and send a request to turn the backlight on for 15 seconds at completion or interruption of moving images shooting.</p> <p>(9) When the mail or message details screen display is specified for the active task with the task switching, an application must resend a request to turn the backlight on depending on the mail or message size.</p>

	<p>(10) During TV phone communication, specify ELIB_LMP_BKLT_TVTALK in the reason for turning the backlight on. In the other TV phone-related operations (e.g., incoming, answer on hold), use a normal voice reason.</p> <p>(11) To send a request to turn the backlight on during TV phone communication, use A-20, (Notifying of Change in Operation Modes,) to notify the LMP management service of the state of TV phone communication beforehand. (See A-20, (Notifying of Change in Operation Modes,) for details.)</p> <p>(12) Before starting camera view mode, specify ELIB_LMP_BKLT_CAMERA_VIEW in the reason for turning the backlight on and send a request of always on. When the state has shifted to camera view mode start or end, use A-20, (Notifying of Change in Operation Modes,) to notify the LMP management service of the state change. (See A-20, (Notifying of Change in Operation Modes,) for details.)</p>
Inhibitions	-
Example of use	<pre> int ret; /* Function return value */ _ELIB_LMP_BKLT_CONFIGSET bklt_cfgset; /* Backlight lighting attribute */ bklt_cfgset.BKLT_Color = ELIB_LMP_BKLT_COLOR1; /* Specify color 1 for the backlight color. */ bklt_cfgset.BKLT_Mode = ELIB_LMP_BKLT_DURING; /* On for a given period of time */ bklt_cfgset.BKLT_KEY_Time = 15; /* On for 15 seconds */ ret = Elib_LMP_Backlight_On ("Application ID",) ELIB_LMP_BKLT_TERMOPEN , &bklt_cfgset); </pre>

8.2 Request to Turn Backlight Off

8.2 Request to Turn Backlight off			
Classification	Function group provided by the LMP management service shared library		
Function name	Request to turn backlight off		
Symbol	Elib_LMP_Backlight_Off		
Format	int Elib_LMP_Backlight_Off (unsigned int ap_id , int reason) ;		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
Reason	int	I	Reason for turning the backlight off Used to solve competitive operation. (See the function explanation for details.)
Return value	Type	I/O	Explanation
Ret	int	O	Normal end: ELIB_LMP_OK Cancelled to prioritize other requests: ELIB_LMP_CANCEL Parameter error: ELIB_LMP_NG
Include file	srv_lmp.h		
Function explanation	<p>This function is used to turn the backlight off.</p> <p>Details of reason (second argument)</p> <p>Specify this argument to determine how the backlight is turned off based on the current lighting status and reason for turning it off when the request is sent. Specify one of the following values in reason.</p> <p>Basically, call this function when an event has occurred, causing the screen display (removal). Specify the reason that corresponds to the event due to which the backlight must be turned off (same as when returning from instantaneous stop).</p> <p>* Normally, message slip, from mobile phone open to closed, power off, and during JAVA operation (KVM) can only be used.</p> <p>ELIB_LMP_BKLT_RCV /* Normal incoming call */ (When the call on hold is received, send a request to turn the backlight on with on for 15 seconds specified.)</p> <p>ELIB_LMP_BKLT_RCVREJECT /* Incoming call rejected */</p> <p>ELIB_LMP_BKLT_RCVREC /* Answer phone */</p> <p>ELIB_LMP_BKLT_TRNSONHOOK /* On-hook outgoing */</p> <p>ELIB_LMP_BKLT_TRNSOFFHOOK /* Off-hook outgoing */</p> <p>ELIB_LMP_BKLT_TERMDISCONN /* Button operation (mobile phone call-ending) */</p> <p>ELIB_LMP_BKLT_OTHERDISCONN /* Disconnected by the remote side */ (Specify this value when the remote side of the incoming call disconnects the call.)</p> <p>ELIB_LMP_BKLT_MSGAUTO /* Automatic message display */</p> <p>ELIB_LMP_BKLT_TRNSTOTALK /* Communication from outgoing */</p>		

	ELIB_LMP_BKLT_RCVTOTALK /* Communication from incoming */ ELIB_LMP_BKLT_RSPWAIT /* Answer on hold */ ELIB_LMP_BKLT_RECDENGON /* Message slip */ ELIB_LMP_BKLT_RECVOICE /* Recording voice slip */ ELIB_LMP_BKLT_RECPLAY /* Playbacking voice slip */ ELIB_LMP_BKLT_SCHEDULE /* Schedule alarm */ ELIB_LMP_BKLT_RCVPACKET /* Incoming packet call */ ELIB_LMP_BKLT_ENDPACKET /* Incoming packet call completed */ ELIB_LMP_BKLT_TERMOPEN /* From mobile phone closed to open */ ELIB_LMP_BKLT_TERMCLOSE /* From mobile phone open to closed */ ELIB_LMP_BKLT_POWERON /* Power on */ ELIB_LMP_BKLT_POWEROFF /* Power off */ ELIB_LMP_BKLT_IMGETCONTENT /* Content collection operation */ ELIB_LMP_BKLT_IMENDCONTENT /* Content collection completed */ ELIB_LMP_BKLT_IMODEMSG /* Mail or message (display or display continuation) */ /* (Read, submenu display, or detailed display of other messages) */ ELIB_LMP_BKLT_IMODEMSGEXIT /* Mail or message display completed (shift to other, etc.) */ ELIB_LMP_BKLT_JAVA /* During JAVA operation (during execution of KVM/JAVA application) */ ELIB_LMP_BKLT_JAM /* During JAVA operation (JAM) */ ELIB_LMP_BKLT_LOWBATTERY /* Low voltage alarm */ ELIB_LMP_BKLT_IMOTION /* At video playback */ ELIB_LMP_BKLT_UDRCV /* At Unrestricted digital data reception */ ELIB_LMP_BKLT_CAMERA /* After shooting still images with the camera function */ ELIB_LMP_BKLT_ACAMERA /* At shooting moving images with the camera function */ ELIB_LMP_BKLT_TVTALK /* TV phone communication started (at screen display) * Change from incoming to communicating, or from outgoing to communicating */ ELIB_LMP_BKLT_CAMERA_VIEW /* Camera view display */ ELIB_LMP_BKLT_ADL_REWRITE /* ADL software update */ ELIB_LMP_BKLT_FAILSAFE /* Fail-safe */ (See (3) in Notes.)
Related message	-
Required procedure	-
Notes	(1) When a request to turn the backlight off is sent for the backlight which is already off, the LMP management service ends normally with the backlight off. (2) When the backlight is on, it may fail to be turned off depending on the reason for turning the backlight off. In this case, the LMP management service returns return value ELIB_LMP_CANCEL. (3) The standby application must specify ELIB_LMP_BKLT_FAILSAFE when the standby screen is displayed before sending a request to turn the backlight off. If

	the backlight comes on regardless of the device state, the LMP management service turns the backlight off. If fail-safe is specified by other applications than standby, the service returns return value ELIB_LMP_NG.
Inhibitions	-
Example of use	ret = Elib_LMP_Backlight_Off("Application ID",) ELIB_LMP_BKLT_TERMCLOSE);

8.3 Request to Turn LED On

8.3 Request to Turn LED On			
Classification	Function group provided by the LMP management service shared library		
Function name	Request to turn LED on		
Symbol	Elib_LMP_LED_On		
Format	int Elib_LMP_LED_On (unsigned int ap_id , int reason, ELIB_LMP_LED_CONFIGSET *led_cfgset);		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
Reason	int	I	Reason for turning the LED on Used to solve competitive operation. (See the function explanation for details.)
led_cfgset	_ELIB_LMP_LED_CONFIGSET *	I	LED lighting attributes Specify the following lighting attributes. (See the function explanation for details.) (1) LED color (2) LED color 2 (3) LED color 3 (4) LED lighting mode (5) LED blinking mode (6) LED blinking cycle on cycle (7) LED blinking cycle off cycle (8) LED lighting time
Return value	Type	I/O	Explanation
Ret	int	O	Normal end: ELIB_LMP_OK Cancelled to prioritize other requests: ELIB_LMP_CANCEL Parameter error: ELIB_LMP_NG
Include file	srv_lmp.h		
Function explanation	<p>This function is used to set the LED lighting attributes and turn the LED on (*1).</p> <p>*1: When this function is called, the incoming LED comes on. Hereinafter, replace the LED with the incoming LED.</p> <p>Details of reason (second argument) Specify this argument to determine how the LED is turned on based on the current lighting status and reason for turning it on when the request is sent (*2).</p> <p>*2: For the function that turns on the LED for setting the incoming illumination tint (RGB adjustment), see A-36, (Request to Turn RGB Adjustment Incoming Illumination</p>		

On: `ELIB_LMP_LEDEMB_RGB_On,`) supported by another interface.

`ELIB_LMP_LED_SELECTCOLOR` /* Incoming illumination or specified illumination */

/* The LED color is being selected for the mail-specified illumination. */

`ELIB_LMP_LED_ABSENTNEWMAIL` /* Checking absence or newly arriving mail */

`ELIB_LMP_LED_CAMERA` /* Still images are being shot with the camera function (when the shutter is pressed or self-timer is used). */

`ELIB_LMP_LED_ACAMERA` /* Moving images are being shot with the camera function (during shooting or when the self-timer is used)*/

`ELIB_LMP_LED_OCR` /* Access reader */

`ELIB_LMP_LED_SDCARD` /* SD card */

Details of `led_cfgset` (third argument)

/* Structure for setting LED lighting attributes */

`typedef struct tag_Elib_LMP_LED_ConfigSet {`

`unsigned short LED_Color1; /* LED color specification 1 */`

`unsigned short LED_Color2; /* LED color specification 2 */`

`unsigned short LED_Color3; /* LED color specification 3 */`

`unsigned short LED_Mode; /* LED lighting mode */`

`unsigned short LED_Blink_Mode; /* LED blinking mode */`

`unsigned short LED_Blink_Cycle; /* LED blinking on cycle */`

`unsigned short LED_Blink_Cycle2; /* LED blinking off cycle */`

`unsigned short LED_Time; /* LED lighting time */`

`} _ELIB_LMP_LED_CONFIGSET;`

(1) LED color: Specifies an LED color.

`ELIB_LMP_LED_COLOR1:` Color 1 (blue/sapphire)

`ELIB_LMP_LED_COLOR2:` Color 2 (color between blue and green/blue > green)

`ELIB_LMP_LED_COLOR3:` Color 3 (color between blue and green/aquamarine)

`ELIB_LMP_LED_COLOR4:` Color 4 (color between blue and green/blue < green)

`ELIB_LMP_LED_COLOR5:` Color 5 (green/emerald)

`ELIB_LMP_LED_COLOR6:` Color 6 (color between green and red/green > red)

`ELIB_LMP_LED_COLOR7:` Color 7 (color between green and red/topaz)

`ELIB_LMP_LED_COLOR8:` Color 8 (color between green and red/green < red)

`ELIB_LMP_LED_COLOR9:` Color 9 (red/ruby)

`ELIB_LMP_LED_COLOR10:` Color 10 (color between blue and red/red > blue)

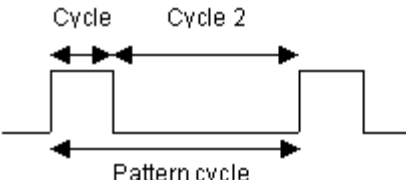
`ELIB_LMP_LED_COLOR11:` Color 11 (color between blue and red/amethyst)

`ELIB_LMP_LED_COLOR12:` Color 12 (white/pearl)

`ELIB_LMP_LED_COLOR13:` Color 13 (gradation) -> Sequentially emitting light of colors 1 to 12 (over a 500-msec cycle)

(2) LED color 2: This value is valid only when `ELIB_LMP_LED_BLINK_ALT` (alternately on) is specified in the blinking mode.

- Colors that can be specified in `LED_Color1` can be specified in LED color 2,

	<p>excepting ELIB_LMP_LED_COLOR13.</p> <p>(3) LED color 3: This value is valid only when ELIB_LMP_LED_BLINK_ALT (alternately on) is specified in the blinking mode. <Alternately on with two colors> - Specify ELIB_LMP_LED_COLOROFF.</p> <p><Alternately on with three colors> - Colors that can be specified in LED_Color 1 can be specified, excepting ELIB_LMP_LED_COLOR13.</p> <p>(4) LED lighting mode: ELIB_LMP_LED_ALWAYS: Always on ELIB_LMP_LED_DURING: On for a given period of time</p> <p>(5) LED blinking mode: This value is invalid when color 13 (gradation) is specified in LED color specification 1. - Colors specified in LED_Color1, LED_Color2, and LED_Color3 glitter alternately. - Currently, only the function of checking absence or newly arriving mail uses this mode. ELIB_LMP_LED_BLINK_OFF: Does not blink ELIB_LMP_LED_BLINK_ON: Blinks ELIB_LMP_LED_BLINK_ALT: Alternately on</p> <p>(6) LED blinking cycle on cycle: When ELIB_LMP_LED_BLINK_ON (Blinks) or ELIB_LMP_LED_BLINK_ALT (Alternately on) is specified in LED_BLINK_Mode, specify the lighting time in seconds using a value from 1 to 65500.</p> <p>(7) LED blinking cycle off cycle: When ELIB_LMP_LED_BLINK_ON (Blinks) or ELIB_LMP_LED_BLINK_ALT (Alternately on) is specified in LED_BLINK_Mode, specify the lights-out time in 10msec using a value from 1 to 65500.</p> <div data-bbox="406 1505 837 1765"> <p>Each blinking</p>  </div> <div data-bbox="874 1568 1508 1742" style="background-color: yellow; border: 1px solid black; padding: 5px;"> <p>Parameter specification method (example)</p> <p>Cycle: 0.5 seconds (500 msec) → LED_Blink_Cycle</p> <p>Cycle 2: 4.5 seconds (10 msec) → LED_Blink_Cycle2</p> <p>* Specify the cycle in units of 10 msec.</p> <p>Pattern cycle: 5 seconds</p> </div> <p>(8) LED lighting time: Specify the LED lighting time in seconds using a value from 1 to 655.</p>
Related message	-
Required	-

procedure	
Notes	<p>(1) The LMP management service determines how the LED is turned on when a request to turn the LED on is sent for the LED which already on. The determination depends on the current lighting status and reason for turning it on when the request is sent.</p> <p>(2) When a request to turn the LED on is sent for the LED which is already on, the LMP management service may determine that the current lighting settings are not updated based on the current lighting status and reason for turning it on. In this case, the service changes nothing (does not update the lighting attributes) and returns return value ELIB_LMP_CANCEL.</p> <p>(3) When alternately on is specified in the blinking mode, LED_Color1, LED_Color2, and LED_Color3 come on alternately over a cycle specified in the blinking cycle. To perform alternately on with two colors, specify ELIB_LMP_LED_COLOROFF in LED_Color3.</p> <p>(4) When a value other than ELIB_LMP_LED_BLINK_ALT is specified in the blinking mode, LED_Color2 and LED_Color3 are invalid. (The setting is not reflected.)</p> <p>(5) When ELIB_LMP_BLINK_OFF is specified in the blinking mode, the blinking cycle is invalid. (The setting is not reflected.)</p> <p>(6) When ELIB_LMP_LED_ALWAYS is specified in the lighting mode, the lighting time is invalid. (The setting is not reflected.)</p> <p>(7) When the gradation is specified in LED_Color1, blinking mode is invalid. (The setting is not reflected.)</p> <p>(8) The sound service turns the LED on in conjunction with the ring tone including voice reception, i-mode mail reception, message R reception, and message F reception.</p> <p>(9) When this function is used to send a request to turn the LED on, the incoming LED comes on using the RGB adjustment value set to the SRAM.</p> <p>(10) Basically, the application that sent a request to turn the incoming LED on must send a request to turn it off.</p> <p>(11) In camera function self-timer operation, an application must send a request to blink the incoming LED according to the remaining number of seconds of the timer for the LMP management service. The blinking cycle for each five seconds or more or less before shooting can be used.</p> <p>(* See the example below for details of how to specify the blinking pattern.)</p> <p>(12) When ELIB_LMP_LED_BLINK_ON is specified in the blinking mode, how to specify the on or off cycle is as follows: Example: When moving images are shot with the camera function (on blinking for 0.5 seconds, off blinking for 4.5 seconds, and blinking cycle of 5 seconds)</p> <p>(13) The LMP management service controls the LED being recharged. Therefore, the application need not consider LED return control from recharge. However, the application can prevent the LMP management service from turning the recharged LED on so that it can keep a series of LED color lighting without being interrupted. To do this, the application must judge whether to prevent the LMP management service from performing LED return processing from recharge and use A-20, (Notifying of Change in Operation Modes,) to inform the service to that effect. (See A-20, (Notifying of Change in Operation Modes,) for details of the function.)</p>
Inhibitions	-
Example of	<pre>short ret; /* Function return value */</pre>

use	<pre>_Elib_LMP_LED_CONFIGSET led_cfgset; /* LED lighting attribute */ led_cfgset. LED_Color = ELIB_LMP_LED_COLOR1; /* Specify color 1 for the LED color. */ led_cfgset. LED_Mode = ELIB_LMP_LED_ALWAYS; /* Always on */ led _cfgset. LED_Blink_Mode = ELIB_LMP_LED_BLINK_ON; /* LED blinking */ led _cfgset. LED_Blink_Cycle = 50; /* The LED blinks over a 500-msec cycle. */ ret = Elib_LMP_LED_On("Application ID",) ELIB_LMP_LED_ON, &led_cfgset) ;</pre>
------------	---

8.4 Request to Turn LED Off

8.4 Request to Turn LED Off			
Classification	Function group provided by the LMP management service shared library		
Function name	Request to turn LED off		
Symbol	Elib_LMP_LED_Off		
Format	int Elib_LMP_LED_Off (unsigned intap_id , int reason) ;		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
Reason	Int	I	Reason for the request to turn the LED off Specifies the reason for turning the LED off. (See the function explanation for details.)
Return value	Type	I/O	Explanation
Ret	Int	O	Normal end: ELIB_LMP_OK Cancelled to prioritize other requests: ELIB_LMP_CANCEL Parameter error: ELIB_LMP_NG
Include file	srv_lmp.h		
Function explanation	<p>This function is used to turn the LED off (*).</p> <p>* When this function is called, the incoming LED goes out.</p> <p>Hereinafter, replace the LED with the incoming LED.</p> <p>Details of reason (second argument)</p> <p>Specify this argument to determine how the LED is turned off based on the current lighting status and reason for turning it off when the request is sent.</p> <p>Specify one of the following values in reason:</p> <p>ELIB_LMP_LED_SELECTCOLOR /* Incoming illumination or specified illumination */</p> <p>* The LED color is being selected for the mail-specified illumination. */</p>		

	<p>ELIB_LMP_LED_ABSENTNEWMAIL /* Checking absence or newly arriving mail */</p> <p>ELIB_LMP_LEDEMB_RGBCOLOR /* The tint of the incoming illumination is within the setting range. *</p> <p>* Tint change (RGB adjustment) */</p> <p>ELIB_LMP_LED_CAMERA /* Still images are being shot with the camera function (when the shutter is pressed or self-timer is used).*/</p> <p>ELIB_LMP_LED_ACAMERA /* Moving images are being shot with the camera function (during shooting or when the self-timer is used)*/</p> <p>ELIB_LMP_LED_OCR /* Access reader */</p> <p>ELIB_LMP_LED_SDCARD /* SD card */</p>
Related message	-
Required procedure	-
Notes	<p>(1) When a request to turn the LED off is sent for the LED which is already off, the LMP management service ends normally with the LED off.</p> <p>(2) When the LED is on, it may fail to be turned off depending on the reason for turning the LED off.</p> <p>In this case, the LMP management service returns return value ELIB_LMP_CANCEL.</p> <p>(3) The LMP management service stops the ring tone to stop the LED in conjunction with the ring tone including voice reception, i-mode mail reception, message R reception, and message F reception.</p> <p>(4) When a request to turn the LED off is sent, the emblem that works in conjunction with the LED also goes out.</p> <p>(5) Basically, the application that sent a request to turn the incoming LED on (that turned the incoming LED on) must send a request to turn it off.</p> <p>(6) The LMP management service controls the LED being recharged. Therefore, the application need not consider recharged LED return control. However, the application can prevent the LMP management service from turning the recharged LED on so that it can keep a series of LED color lighting without being interrupted. To do this, the application must judge whether to prevent the LMP management service from performing LED return processing from recharge and use A-20,</p>

	(Notifying of Change in Operation Modes,) to inform the service to that effect. (See A-20, (Notifying of Change in Operation Modes,) for details of the function.)
Inhibitions	-
Example of use	<pre>int ret; /* Function return value */ ret = Elib_LMP_LED_Off ("Application ID",) ELIB_LMP_LED_OFF);</pre>

8.5 Referencing Backlight Lighting state

8-5 Referencing Backlight Lighting State			
Classification	Function group provided by the LMP management service shared library		
Function name	Referencing backlight lighting state		
Symbol	Elib_LMP_Backlight_GetSts		
Format	int Elib_LMP_Backlight_GetSts (unsigned int ap_id) ;		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
Return value	Type	I/O	Explanation
Ret	int	O	Normal end:
			When the backlight is off: ELIB_LMP_BKLT_OFF
			When the backlight is on: ELIB_LMP_BKLT_ON
			Abnormal end: ELIB_LMP_NG
Include file	srv_lmp.h		
Function explanation	This function is used to reference the backlight lighting state.		
Related message	-		
Required procedure	-		
Notes	-		
Inhibitions	-		
Example of use	<pre> int ret; ret = Elib_LMP_Backlight_GetSts ("Application ID"); if(ret == ELIB_LMP_BKLT_ON) { /* When the backlight is on */ } else if(ret == ELIB_LMP_BKLT_OFF) { /* When the backlight is off */ } </pre>		

8.6 Referencing LED Lighting State

8-6 Referencing LED Lighting State			
Classification	Function group provided by the LMP management service shared library		
Function name	Referencing LED lighting state		
Symbol	Elib_LMP_LED_GetSts		
Format	int Elib_LMP_LED_GetSts (unsigned int ap_id) ;		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
Return value	Type	I/O	Explanation
Ret	int	O	Normal end: When the LED is off: ELIB_LMP_LED_OFF When the LED is on: ELIB_LMP_LED_ON Abnormal end: ELIB_LMP_NG
Include file	srv_imp.h		
Function explanation	This function is used to reference the LED lighting state. This function returns the LED lighting state with the return value.		
Related message	-		
Required procedure	-		
Notes	-		
Inhibitions	-		
Example of use	<pre>int ret; ret = Elib_LMP_LED_GetSts ("Application ID"); if(ret == ELIB_LMP_LED_ON) { /* When the LED is on */ }</pre>		

```
} else if(ret == ELIB_LMP_BKLT_OFF) {
```

```
    /* When the LED is off */
```

```
}
```

8.7 Listing LMP Titles and Collecting Color Numbers

8-7 Listing LMP Titles and Collecting Color Numbers			
Classification	Function group provided by the LMP management service shared library		
Function name	Listing LMP titles and collecting color numbers		
Symbol	Elib_LMP_Get_Information		
Format	int Elib_LMP_Get_Information (unsigned int ap_id , int mode, int number, _ELIB_LMP_TITLELIST *titlelist) ;		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
Mode	Int	I	Specifies the type of information to be collected. (See the function explanation for details.)
Number	Int	I	Specifies the number of the color title to be collected.
Titlelist	_ELIB_LMP_TITLELIST *	O	Specifies the array of the structure _ELIB_LMP_TITLELIST that stores the list of color titles. (See the function explanation for details.) (1) Number of the incoming illumination color (2) Flag that indicates whether the color title can be changed (3) Size of the color title character string (4) Color title character string
Return value	Type	I/O	Explanation
Ret	Int	O	Normal end: Number of titles stored Parameter error: ELIB_LMP_NG
Include file	srv_lmp.h		
Function explanation	<p>This function is used to collect the following information for the LMP specified in mode (second argument):</p> <ul style="list-style-type: none"> (1) List of color titles for the backlight, LED, rear emblem, and incoming illumination (2) Color numbers that correspond to each color (3) Information that indicates that the corresponding color can be changed 		

Details of mode (second argument)

```
ELIB_LMP_ILLDATA          /* Collecting information about the incoming
illumination */
```

* SYNPHONIA supports ELIB_LMP_ILLDATA only.

Details of number (third argument)

This argument stores a number (1 to 13) of the color to be collected.

Details of titlelist (fourth argument)

```
/* Structure that stores the color title */
```

```
typedef struct tag_ELIB_LMP_TITLELIST{

    int    colornumber;      /* Number of the incoming illumination color *
                               * A number from ELIB_LMP_LED_COLOR1 to
13 */

    int    colorflag;        /* Flag that indicates whether the color title can be
changed */

    int    length            /* Size of the color title character string */

    unsigned char  title[20]; /* Color title character string */

}_ELIB_LMP_TITLELIST ;
```

(1) Number of the incoming illumination color:

Specifies a color number from ELIB_LMP_LED_COLOR1 to 13 for the incoming illumination.

(2) Flag that indicates whether the color title can be changed: Indicates whether

	<p>the color title can be changed.</p> <p>ELIB_LMP_FIXEDCOLOR: Fixed color (cannot be changed)</p> <p>ELIB_LMP_ORIGINALCOLOR: Original color (can be changed)</p> <p>(3) Size of the color title character string: Stores the size of the color title character string.</p> <p>(4) Color title character string: Stores the color title character string.</p>
Related message	-
Required procedure	-
Notes	<p>(1) When a color title number from 1 to 12 is specified in number (third argument), the original color (that can be changed) is specified in the (flag that indicates whether the color title can be changed) of the structure that stores the color titles.</p> <p>(2) When color title number 13 (gradation) is specified in number (third argument), the fixed color (that cannot be changed) is specified in the (flag that indicates whether the color title can be changed) of the structure that stores the color titles.</p> <p>(3) When OFF or a color title number other than 1 to 13 is specified in number (third argument), a parameter error is assumed.</p>
Inhibitions	-
Example of use	<pre> _ELIB_LMP_TITLELIST led_title; int ret ; ret = Elib_LMP_Get_Information("Application ID",) ELIB_LMP_ILLDATA, 1, &led_title); </pre>

8.8 Referncing Original LMP Colors

8-8 Referencing Original LMP Colors			
Classification	Function group provided by the LMP management service shared library		
Function name	Referencing original LMP colors		
Symbol	Elib_LMP_Get_ColorData		
Format	int Elib_LMP_Get_ColorData (unsigned int ap_id , int mode, int colornumber, _ELIB_LMP_COLORDATA * get_data) ;		
Argument	Type	I/O	Explanation
ap_id	Unsigned int	I	Application ID
Mode	Int	I	Information type Specifies the type of information to be collected. (See the function explanation for details.)
colornumber	Int	I	Color number Specifies the color numbers for the LED, backlight, and incoming illumination whose colors are to be inquired. (See the function explanation for details.)
get_data	_ELIB_LMP_COLORDATA *	O	Pointer to the original color data structure. (See the function explanation for details.) (1) Color number (not used) (2) RGB luminance information storage array
Return value	Type	I/O	Explanation
Ret	Int	O	Normal end: Color numbers of the stored titles Parameter error: ELIB_LMP_NG
Include file	srv_lmp.h		
Function explanation	<p>This function is used to collect data of the original colors for the backlight, LED, rear emblem, and incoming illumination that can be changed by the user.</p> <p>Details of mode (second argument)</p> <p>ELIB_LMP_ILLDATA /* Collecting information about the incoming illumination */</p>		

Details of colornumber (third argument)

Specify the numbers of the colors for the LED, backlight, and incoming illumination in colornumber.

- When ELIB_LMP_ILLDATA is specified in mode (second argument), specify one of the following values:

ELIB_LMP_LED_COLOR1: Color 1 (blue)

ELIB_LMP_LED_COLOR2: Color 2 (color between blue and green/blue > green)

ELIB_LMP_LED_COLOR3: Color 3 (color between blue and green)

ELIB_LMP_LED_COLOR4: Color 4 (color between blue and green/blue < green)

ELIB_LMP_LED_COLOR5: Color 5 (green)

ELIB_LMP_LED_COLOR6: Color 6 (color between green and red/green > red)

ELIB_LMP_LED_COLOR7: Color 7 (color between green and red)

ELIB_LMP_LED_COLOR8: Color 8 (color between green and red/green < red)

ELIB_LMP_LED_COLOR9: Color 9 (red)

ELIB_LMP_LED_COLOR10: Color 10 (color between red and blue/red > blue)

ELIB_LMP_LED_COLOR11: Color 11 (color between red and blue)

ELIB_LMP_LED_COLOR12: Color 12 (white)

Details of get_data (fourth argument)

This structure stores information about the luminance of the backlight, LED, and incoming illumination.

/* Pointer to the original color data structure */

	<pre>typedef struct tag_ELIB_LMP_COLORDATA { unsigned short lampnum; /* Number of the lamp that can be adjusted (not used)*/ unsigned char lamp[6]; /* Array for storing luminance information for each lamp */ } _ELIB_LMP_COLORDATA; * The array for storing luminance information for each lamp is defined as follows: lamp[0]: Red luminance information (tint) ELIB_LMP_RGB_RCOLOR1 to ELIB_LMP_RGB_RCOLOR12 lamp[1]: Green luminance information (tint) ELIB_LMP_RGB_GCOLOR1 to ELIB_LMP_RGB_GCOLOR12 lamp[2]: Blue luminance information (tint) ELIB_LMP_RGB_BCOLOR1 to ELIB_LMP_RGB_BCOLOR12 lamp[3] to lamp[5]: Not used</pre>
Related message	-
Required procedure	-
Notes	-
Inhibitions	-
Example of use	<pre>_ELIB_LMP_COLORDATA get_data; int ret ; ret = Elib_LMP_Get_ColorData ("Application ID",) ELIB_LMP_ILLDATA, ELIB_LMP_LED_COLOR1, &get_data);</pre>

8.9 Setting Original LMP Colors

8-9 Setting Original LMP Colors			
Classification	Function group provided by the LMP management service shared library		
Function name	Setting original LMP colors		
Symbol	Elib_LMP_Set_ColorData		
Format	int Elib_LMP_Set_ColorData (unsigned int ap_id , int mode, int colornumber, _ELIB_LMP_COLORDATA * set_data) ;		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
Mode	Int	I	Information type Specifies the type of information to be collected. (See the function explanation for details.)
colornumber	Int	I	Color number Specifies the color numbers for the LED, backlight, and incoming illumination (incoming LED and emblem) whose colors are to be inquired. (See the function explanation for details.)
set_data	_ELIB_LMP_COLORDATA *	I	Pointer to the original color data structure. (See the function explanation for details.) (1) Color number (not used) (2) luminance information storage array
Return value	Type	I/O	Explanation
Ret	Int	O	Normal end: Color numbers of the stored titles Parameter error: ELIB_LMP_NG
Include file	srv_lmp.h		
Function explanation	<p>This function is used to specify data of the original colors for the backlight, LED, rear emblem, and incoming illumination that can be changed by the user.</p> <p>Details of mode (second argument)</p> <p>ELIB_LMP_ILLDATA /* Collecting information about the incoming illumination */</p>		

Details of colornumber (third argument)

- When ELIB_LMP_ILLDATA is specified in mode (second argument), specify one of the following values:

ELIB_LMP_LED_COLOR1: Color 1 (blue)

ELIB_LMP_LED_COLOR2: Color 2 (color between blue and green/blue > green)

ELIB_LMP_LED_COLOR3: Color 3 (color between blue and green)

ELIB_LMP_LED_COLOR4: Color 4 (color between blue and green/blue < green)

ELIB_LMP_LED_COLOR5: Color 5 (green)

ELIB_LMP_LED_COLOR6: Color 6 (color between green and red/green > red)

ELIB_LMP_LED_COLOR7: Color 7 (color between green and red)

ELIB_LMP_LED_COLOR8: Color 8 (color between green and red/green < red)

ELIB_LMP_LED_COLOR9: Color 9 (red)

ELIB_LMP_LED_COLOR10: Color 10 (color between red and blue/red > blue)

ELIB_LMP_LED_COLOR11: Color 11 (color between red and blue)

ELIB_LMP_LED_COLOR12: Color 12 (white)

Details of set_data (fourth argument)

This structure specifies the luminance of the backlight, LED, and incoming illumination.

/* Pointer to the original color data structure */

typedef struct tag_ELIB_LMP_COLORDATA {

 unsigned short lampnum; /* Number of the lamp that can be adjusted
(not used) */

	<pre> unsigned char lamp[6]; /* Array for storing luminance information for each lamp */ } _ELIB_LMP_COLORDATA; * The array for storing luminance information for each lamp is defined as follows: lamp[0]: Red luminance information (tint) ELIB_LMP_RGB_RCOLOR1 to ELIB_LMP_RGB_RCOLOR12 lamp[1]: Green luminance information (tint) ELIB_LMP_RGB_GCOLOR1 to ELIB_LMP_RGB_GCOLOR12 lamp[2]: Blue luminance information (tint) ELIB_LMP_RGB_BCOLOR1 to ELIB_LMP_RGB_BCOLOR12 lamp[3] to lamp[5]: Not used </pre>
Related message	-
Required procedure	-
Notes	-
Inhibitions	-
Example of use	<pre> _ELIB_LMP_COLORDATA set_data; int ret ; ret = Elib_LMP_Set_ColorData ("Application ID",) ELIB_LMP_ILLDATA, ELIB_LMP_LED_COLOR1, &set_data) ; </pre>

8.10 Changing Original LMP Color Titles

8-10 Changing Original LMP Color Titles			
Classification	Function group provided by the LMP management service shared library		
Function name	Changing original LMP color titles		
Symbol	Elib_LMP_Set_ColorTitle		
Format	<pre> int Elib_LMP_Set_ColorTitle (unsigned int ap_id , int mode, int colornumber, int length , unsigned char *org_title) ; </pre>		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
Mode	int	I	Information type Specifies the type of information to be collected. (See the function explanation for details.)
colornumber	int	I	Color number (See the function explanation for details.)
Length	int	I	Size of the character string
org_title	unsigned char *	I	Pointer that indicates the character string of the original color title. (See the function explanation for details.)
Return value	Type	I/O	Explanation
Ret	int	O	Normal end: Color numbers of the stored titles Parameter error: ELIB_LMP_NG
Include file	srv_lmp.h		
Function explanation	<p>This function is used to change the original color titles for the backlight, LED, rear emblem, and incoming illumination that can be changed by the user.</p> <p>Details of mode (second argument): Specify the following value in mode:</p> <p>ELIB_LMP_ILLDATA /* Collecting information about the incoming illumination */</p> <p>Details of colornumber (third argument): Specify the numbers of the colors for the LED, backlight, and incoming illumination in colornumber.</p> <p>- When ELIB_LMP_ILLDATA is specified in mode (second argument), specify one of the following values:</p>		

	<p>ELIB_LMP_LED_COLOR1: Color 1 (blue)</p> <p>ELIB_LMP_LED_COLOR2: Color 2 (color between blue and green/blue > green)</p> <p>ELIB_LMP_LED_COLOR3: Color 3 (color between blue and green)</p> <p>ELIB_LMP_LED_COLOR4: Color 4 (color between blue and green/blue < green)</p> <p>ELIB_LMP_LED_COLOR5: Color 5 (green)</p> <p>ELIB_LMP_LED_COLOR6: Color 6 (color between green and red/green > red)</p> <p>ELIB_LMP_LED_COLOR7: Color 7 (color between green and red)</p> <p>ELIB_LMP_LED_COLOR8: Color 8 (color between green and red)/green < red)</p> <p>ELIB_LMP_LED_COLOR9: Color 9 (red)</p> <p>ELIB_LMP_LED_COLOR10: Color 10 (color between red and blue/red > blue)</p> <p>ELIB_LMP_LED_COLOR11: Color 11 (color between red and blue)</p> <p>ELIB_LMP_LED_COLOR12: Color 12 (white)</p> <p>Details of org_title (fifth argument):</p> <p>This argument is a pointer that indicates the character strings of the original color titles for the backlight, LED, and incoming illumination to be overwritten.</p> <p>The character string must be null-ended. Up to 20 bytes can be used for the character string.</p>
Related message	-
Required procedure	-
Notes	-
Inhibitions	-
Example of use	<pre>int ret ; char str[20] ; strcpy(str, (test), 4) ; ret = Elib_LMP_Set_ColorTitle("Application ID",) ELIB_LMP_ILLDATA, ELIB_LMP_LED_COLOR1, strlen(str), str) ;</pre>

8.11 Referencing LMP Setting

8-11 Referencing LMP Settings			
Classification	Function group provided by the LMP management service shared library		
Function name	Referencing LMP settings		
Symbol	Elib_LMP_GetBKLTMode		
Format	int Elib_LMP_GetBKLTMode (unsigned int ap_id , int func_id);		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
func_id	Int	I	Function number Specifies the function whose setting state is to be updated. (See the function explanation for details.)
Return value	Type	I/O	Explanation
Ret	Int	O	Normal end: The setting value saved to the SRAM is returned. (See the function explanation for details.) Parameter error: ELIB_LMP_NG
Include file	srv_lmp.h		

<p>Function explanation</p>	<p>This function is used to reference the settings of backlight display, display of the backlight being recharged, backlight lighting range, and backlight brightness.</p> <p>Details of func_id (second argument): Specify one of the following values in func_id:</p> <p>ELIB_LMP_BKLTSET: Setting of backlight display</p> <p>ELIB_LMP_CHGMODESET: Setting of display of the backlight being recharged</p> <p>ELIB_LMP_BKLTRANGE: Setting of backlight lighting range</p> <p>ELIB_LMP_BKLTVOL: Setting of backlight brightness</p> <p>ELIB_LMP_GLIMMERSET: Power-saving setting</p> <p>ELIB_LMP_GLIMMERTIME: Setting of power-saving wait time</p> <p>ELIB_LMP_MOVBKLTSET: Setting of TV phone illumination</p> <p>Details of the return value: Return value ret depends on the value specified in func_id (second argument). One of the following values is used:</p> <ul style="list-style-type: none"> - When ELIB_LMP_BKLTSET is specified in func_id: <p>ELIB_LMP_BKLTSET_ON: Setting of backlight display on</p> <p>ELIB_LMP_BKLTSET_OFF: Setting of backlight display off</p> - When ELIB_LMP_CHGMODESET is specified in func_id: <p>ELIB_LMP_CHGMODESET_ON: Always on</p> <p>ELIB_LMP_CHGMODESET_OFF: Standard</p> - When ELIB_LMP_BKLTRANGE is specified in func_id: <p>ELIB_LMP_ALL: LCD + button</p> <p>ELIB_LMP_LCD: LCD only</p> - When ELIB_LMP_BKLTVOL is specified in func_id: <p>ELIB_LMP_BKLTSET_ON1: Level 1</p> <p>ELIB_LMP_BKLTSET_ON2: Level 2</p> <p>ELIB_LMP_BKLTSET_ON3: Level 3</p> - When ELIB_LMP_GLIMMERSET is specified in func_id: <p>ELIB_LMP_GLIMMERSET_ON: Setting of power-saving start on</p> <p>ELIB_LMP_GLIMMERSET_OFF: Setting of power-saving start off</p> - When ELIB_LMP_GLIMMERTIME is specified in func_id: <p>2 to 20: Power-saving wait time (minute)</p> - When ELIB_LMP_MOVBKLTSET is specified in func_id: <p>ELIB_LMP_BKLTSET_STD: Standard</p> <p>ELIB_LMP_BKLTSET_NOW: Always on</p>
------------------------------------	--

Related message	-
Required procedure	-
Notes	-
Inhibitions	-
Example of use	<pre>int ret; /* Reference the state of the backlight display setting. */ ret = Elib_LMP_GetBKLTMode ("Application ID",) ELIB_LMP_BKLTSET) ;</pre>

8.12 Referencing LMP Settings

8-12 Registering LMP Settings			
Classification	Function group provided by the LMP management service shared library		
Function name	Registering LMP settings		
Symbol	Elib_LMP_SetBKLTMode		
Format	int Elib_LMP_SetBKLTMode (unsigned int ap_id , int func_id, int set_val) ;		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
func_id	int	I	Function number Specifies the function whose setting state is to be updated. (See the function explanation for details.)
set_val	int	I	Setting value Specifies the setting value to be saved to the SRAM. (See the function explanation for details.)
Return value	Type	I/O	Explanation
Ret	int	O	Normal end: ELIB_LMP_OK Parameter error: ELIB_LMP_NG
Include file	srv_imp.h		
Function explanation	<p>This function is used to update the settings of backlight display, display of the backlight being recharged, backlight lighting range, and backlight brightness.</p> <p>Details of func_id (second argument): Specify one of the following values in func_id:</p> <p>ELIB_LMP_BKLTSET: Setting of backlight display</p> <p>ELIB_LMP_CHGMODESET: Setting of display of the backlight being recharged</p> <p>ELIB_LMP_BKLTRANGE: Setting of backlight lighting range</p> <p>ELIB_LMP_BKLTVOL: Setting of backlight brightness</p> <p>ELIB_LMP_GLIMMERSET: Power-saving setting</p> <p>ELIB_LMP_GLIMMERTIME: Setting of power-saving wait time</p>		

ELIB_LMP_MOVBKLTSET: Setting of TV phone illumination

Details of set_val (third argument): Specify one of the following values in set_val:

- When ELIB_LMP_BKLTSET is specified in func_id:

ELIB_LMP_BKLTSET_ON: On

ELIB_LMP_BKLTSET_OFF: Off

- When ELIB_LMP_CHGMODESET is specified in func_id:

ELIB_LMP_CHGMODESET_ON: Always on

ELIB_LMP_CHGMODESET_OFF: Standard

- When ELIB_LMP_BKLTRANGE is specified in func_id:

ELIB_LMP_ALL: LCD + button

ELIB_LMP_LCD: LCD only

- When ELIB_LMP_BKLTVOL is specified in func_id:

ELIB_LMP_BKLTSET_ON1: Level 1

ELIB_LMP_BKLTSET_ON2: Level 2

ELIB_LMP_BKLTSET_ON3: Level 3

- When ELIB_LMP_GLIMMERSET is specified in func_id:

ELIB_LMP_GLIMMERSET_ON: On

ELIB_LMP_GLIMMERSET_OFF: Off

	<ul style="list-style-type: none"> - When ELIB_LMP_GLIMMERTIME is specified in func_id: 2 to 20: 2 to 20 minutes - When ELIB_LMP_MOVBKLTSET is specified in func_id: ELIB_LMP_BKLTSET_STD: Standard ELIB_LMP_BKLTSET_NOW: Always on
Related message	-
Required procedure	-
Notes	-
Inhibitions	-
Example of use	<pre>int ret; /* Turn the state of the backlight display setting on. */ ret = Elib_LMP_GetBKLTMode ("Application ID", ELIB_LMP_BKLTSET, ELIB_LMP_BKLTSET_ON);</pre>

8.13 Referencing LMP Incoming Illumination Settings

8-13 Referencing LMP Incoming Illumination Settings			
Classification	Function group provided by the LMP management service shared library		
Function name	Referencing LMP incoming illumination settings		
Symbol	Elib_LMP_GetLEDColor		
Format	int Elib_LMP_GetLEDColor (unsigned int ap_id , int rcvmode) ;		
Argument	Type	I/O	Explanation
ap_id	Unsigned int	I	Application ID
Rcvmode	Int	I	Incoming type Specifies the type of the incoming call or mail whose setting state is to be referenced. (See the function explanation for details.)
Return value	Type	I/O	Explanation
Ret	Int	O	Normal end: The setting value saved to the SRAM is returned. (See the function explanation for details.) Parameter error: ELIB_LMP_NG
Include file	srv_lmp.h		
Function explanation	<p>This function is used to reference the colors specified for the incoming illumination.</p> <p>Details of rcvmode (second argument) Specify the type of the incoming call or mail to be referenced in rcvmode.</p> <p>ELIB_LMP_VOICERECV: Voice reception ELIB_LMP_IMODEMAIL: i-mode mail ELIB_LMP_IMODEMSG_R: Message R ELIB_LMP_IMODEMSG_F: Message F ELIB_LMP_TVVOICERECV: TV phone reception</p> <p>Return value ret depends on the setting state of the incoming type. One of the following values is used:</p> <p>ELIB_LMP_LED_COLOR1: Color 1 (blue) ELIB_LMP_LED_COLOR2: Color 2 (color between blue and green/blue > green) ELIB_LMP_LED_COLOR3: Color 3 (color between blue and green) ELIB_LMP_LED_COLOR4: Color 4 (color between blue and green/blue < green) ELIB_LMP_LED_COLOR5: Color 5 (green)</p>		

	<p>ELIB_LMP_LED_COLOR6: Color 6 (color between green and red/green > red)</p> <p>ELIB_LMP_LED_COLOR7: Color 7 (color between green and red)</p> <p>ELIB_LMP_LED_COLOR8: Color 8 (color between green and red/green < red)</p> <p>ELIB_LMP_LED_COLOR9: Color 9 (red)</p> <p>ELIB_LMP_LED_COLOR10: Color 10 (color between red and blue/red > blue)</p> <p>ELIB_LMP_LED_COLOR11: Color 11 (color between red and blue)</p> <p>ELIB_LMP_LED_COLOR12: Color 12 (white)</p> <p>ELIB_LMP_LED_COLOR13: Color 13 (gradation) -> Sequentially emitting light of colors 1 to 12 (over a 500-msec cycle)</p> <p>Return values depending on the setting state of the incoming type</p> <p>Voice reception: Colors 1 to 13</p> <p>i-mode mail: Colors 1 to 13</p> <p>Message R: Colors 1 to 13</p> <p>Message F: Colors 1 to 13</p> <p>TV phone reception: Colors 1 to 13</p>
Related message	-
Required procedure	-
Notes	-
Inhibitions	-
Example of use	<pre>int ret; /* Reference the illumination color at voice reception. */ ret = Elib_LMP_GetLEDColor ("Application ID", ELIB_LMP_VOICERECV);</pre>

8.14 Referencing LMP Incoming Illumination Setting

8-14 Registering LMP Incoming Illumination Settings			
Classification	Function group provided by the LMP management service shared library		
Function name	Registering LMP incoming illumination settings		
Symbol	Elib_LMP_SetLEDColor		
Format	int Elib_LMP_SetLEDColor (unsigned int ap_id , int rcvmode, int setcolor) ;		
Argument	Type	I/O	Explanation
ap_id	Unsigned int	I	Application ID
Rcvmode	Int	I	Incoming type Specifies the type of the incoming call or mail whose illumination color is to be registered. (See the function explanation for details.)
Setcolor	Int	I	Setting value Specifies the color of the illumination to be saved to the SRAM. (See the function explanation for details.)
Return value	Type	I/O	Explanation
Ret	Int	O	Normal end: ELIB_LMP_OK Parameter error: ELIB_LMP_NG
Include file	srv_lmp.h		
Function explanation	<p>This function is used to register the colors specified for the incoming illumination.</p> <p>Details of rcvmode (second argument): Specify the type of the incoming call or mail to be referenced in rcvmode. ELIB_LMP_VOICERECV: Voice reception ELIB_LMP_IMODEMAIL: i-mode mail ELIB_LMP_IMODEMSG_R: Message R ELIB_LMP_IMODEMSG_F: Message F ELIB_LMP_TVVOICERECV: TV phone reception</p> <p>Details of setcolor (third argument) Specify the color of the incoming illumination to be registered in setcolor. Specify one of the following values: ELIB_LMP_LED_COLOR1: Color 1 (blue) ELIB_LMP_LED_COLOR2: Color 2 (color between blue and green/blue > green) ELIB_LMP_LED_COLOR3: Color 3 (color between blue and green) ELIB_LMP_LED_COLOR4: Color 4 (color between blue and green/blue < green) ELIB_LMP_LED_COLOR5: Color 5 (green)</p>		

	ELIB_LMP_LED_COLOR6: Color 6 (color between green and red/green > red) ELIB_LMP_LED_COLOR7: Color 7 (color between green and red) ELIB_LMP_LED_COLOR8: Color 8 (color between green and red/green < red) ELIB_LMP_LED_COLOR9: Color 9 (red) ELIB_LMP_LED_COLOR10: Color 10 (color between red and blue/red > blue) ELIB_LMP_LED_COLOR11: Color 11 (color between red and blue) ELIB_LMP_LED_COLOR12: Color 12 (white) ELIB_LMP_LED_COLOR13: Color 13 (gradation) -> Sequentially emitting light of colors 1 to 12 (over a 500-msec cycle)
Related message	-
Required procedure	-
Notes	-
Inhibitions	-
Example of use	<pre> int ret; /* Reference the illumination color at voice reception. */ ret = Elib_LMP_SetLEDColor("Application ID", ELIB_LMP_VOICERECV, ELIB_LMP_LED_COLOR1) ; </pre>

8.15 Request to Turn Rear Emblem On

8-15 Request to Turn Rear Emblem On			
Classification	Function group provided by the LMP management service shared library		
Function name	Request to turn rear emblem on		
Symbol	Elib_LMP_Emblem_On		
Format	<pre>int Elib_LMP_Emblem_On (unsigned int ap_id , int reason, int condition ,_ELIB_LMP_EMB_CONFI GSET *emb_cfgset);</pre>		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
reason	Int	I	Reason for turning the rear emblem on Used to solve competitive operation. (See the function explanation for details.)
condition	Int	I	Condition for turning the rear emblem on Specifies the condition for turning the rear emblem on. (See the function explanation for details.)
emb_cfgset	_ELIB_LMP_EMB_CONFIGSET *	I	ENB lighting attributes. (See the function explanation for details.) (1) Lighting color 1 (2) Lighting color 2 (3) Lighting mode (4) Blinking mode (5) Blinking pattern (6) Lighting time
Return value	Type	I/O	Explanation
ret	Int	O	Normal end: ELIB_LMP_OK Cancelled to prioritize other requests: ELIB_LMP_CANCEL Parameter error: ELIB_LMP_NG
Include file	srv_imp.h		
Function explanation	<p>This function is used to set the lighting attributes of the rear emblem and turn the rear emblem on (*).</p> <p>* When this function is called, the incoming LED comes on. Hereinafter, replace the rear emblem with the incoming LED.</p> <ul style="list-style-type: none"> - The LMP management service determines how the emblem is turned on when a request to turn the emblem on is sent for the emblem which is already on. The determination depends on the current lighting status and reason for turning it on when the request is sent. 		

- If the LMP management service determines that the current lighting settings are not updated, it changes nothing (does not update the lighting attributes). In this case, the service returns return value ELIB_LMP_CANCEL.

Details of reason (second argument)

Specify this argument to determine how the rear emblem is turned on based on the current lighting status and reason for turning it on when the request is sent.

Specify either of the following values in reason:

ELIB_LMP_EMB_SELECTEMB: The setting of the communicating illumination is being selected.

ELIB_LMP_EMB_TALK: Communicating. (* Only the telephone application can use this value.)

Details of condition (third argument)

This argument specifies whether the rear emblem is turned on unconditionally or in reference to the SRAM settings.

ELIB_LMP_EMB_DEPENDENT: In reference to settings

ELIB_LMP_EMB_INDEPENDENT: Unconditionally

Details of emb_cfgset (fourth argument)

/* Structure for setting rear emblem lighting attributes */

```
typedef struct tag_Elib_LMP_EMB_ConfigSet {
```

```
    unsigned short  EMB_Color1;           /* Lighting color 1 (always valid) */
```

```
    unsigned short EMB_Color2;           /* Not used */
```

```
    unsigned short EMB_Mode;             /* Lighting mode */
```

```
    unsigned short EMB_Blink_Mode;       /* Not used */
```

```
    unsigned short EMB_Blink_Pattern ;   /* Blinking pattern */
```

```
    unsigned short EMB_Time ;            /* Lighting time */
```

```
} _ELIB_LMP_EMB_CONFIGSET;
```

(1) Lighting color 1:

ELIB_LMP_ILLUM_COLOR1: Communicating color 1 (blue)

ELIB_LMP_ILLUM_COLOR2: Communicating color 2 (color between blue and green)

ELIB_LMP_ILLUM_COLOR3: Communicating color 3 (green)

ELIB_LMP_ILLUM_COLOR4: Communicating color 4 (color between green and red)

ELIB_LMP_ILLUM_COLOR5: Communicating color 5 (red)

ELIB_LMP_ILLUM_COLOR6: Communicating color 6 (color between red and blue)

ELIB_LMP_ILLUM_COLOR7: Communicating color 7 (white)

ELIB_LMP_ILLUM_COLOR8: Gradation pattern 1

ELIB_LMP_ILLUM_COLOR9: Gradation pattern 2

ELIB_LMP_ILLUM_COLOR10: Gradation pattern 3

(2) Lighting color 2: Not used

(3) Lighting mode:

ELIB_LMP_EMB_ALWYAS: Always on

ELIB_LMP_EMB_DURING: On for a given period of time

(4) Blinking mode: Not used

	<p>(5) Blinking pattern: ELIB_LMP_EMB_PATTERN1: Pattern indicating that the mobile phone is on hold under communication ELIB_LMP_EMB_PATTERN2: Pattern indicating communicating</p> <p>(6) Lighting time: Lighting time (Specify the time in seconds using a value from 1 to 655.)</p>
Related message	-
Required procedure	-
Notes	The LMP management service stops the ring tone to stop the emblem in conjunction with the ring tone including voice reception, i-mode mail reception, message R reception, and message F reception.
Inhibitions	-
Example of use	<pre> int ret; /* Function return value */ _Elib_LMP_EMB_CONFIGSET emb_cfgset; /* Emblem lighting attribute */ emb_cfgset.EMB_Color1 = ELIB_LMP_EMB_COLOR1; /* Color 1 is used for the emblem. */ emb_cfgset.EMB_Mode = ELIB_LMP_EMB_ALWAYS; /* Always on */ emb_cfgset.EMB_Blink_Pattern = ELIB_LMP_EMB_PATTERN1; /* Blinking with a fixed pattern */ ret = Elib_LMP_Emblem_On ("Application ID", /* Function setting application */ ELIB_LMP_EMB_SELECTEMB, /* The lighting reason is that the rear emblem settings are being selected. */ ELIB_LMP_EMB_INDEPENDENT, /* Unconditional lighting is used for the lighting condition. */ &emb_cfgset); </pre>

8.16 Request to Turn Rear Emblem Off

8-16 Request to Turn Rear Emblem Off			
Classification	Function group provided by the LMP management service shared library		
Function name	Request to turn rear emblem off		
Symbol	Elib_LMP_Emblem_Off		
Format	int Elib_LMP_Emblem_Off (unsigned int ap_id , int reason);		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
reason	Int	I	Reason for turning the rear emblem off Used to solve competitive operation. (See the function explanation for details.)
Return value	Type	I/O	Explanation
ret	Int	O	Normal end: ELIB_LMP_OK Cancelled to prioritize other requests: ELIB_LMP_CANCEL Parameter error: ELIB_LMP_NG
Include file	srv_lmp.h		
Function explanation	<p>This function is used to turn the rear emblem off (*).</p> <p>* When this function is called, the incoming LED goes out.</p> <p>Hereinafter, replace the rear emblem with the incoming LED.</p> <ul style="list-style-type: none"> - When a request to turn the emblem off is sent for the emblem which is already off, the LMP management service ends normally with the emblem off. - When the emblem is on, it may fail to be turned off depending on the reason for turning the emblem off. In this case, the LMP management service returns return value ELIB_LMP_CANCEL. <p>Details of reason (second argument)</p> <p>Specify this argument to determine how the rear emblem is turned off based on the current lighting status and reason for turning it off when the request is sent.</p> <p>Specify either of the following values in reason:</p> <p>ELIB_LMP_EMB_SELECTEMB: The setting of the communicating illumination is being selected.</p> <p>ELIB_LMP_EMB_TALK: Communicating. (* Only the telephone application can use this value.)</p>		
Related message	-		
Required procedure	-		

Notes	The LMP management service stops the ring tone to stop the emblem in conjunction with the ring tone including voice reception, i-mode mail reception, message R reception, and message F reception.
Inhibitions	-
Example of use	<pre>int ret; /* Function return value */ ret = Elib_LMP_Emblem_Off("Application ID", ELIB_LMP_EMB_SELECTEMB);</pre>

8.17 Referencing incoming LED-associated Settings

8-17 Referencing Incoming LED-associated Settings			
Classification	Function group provided by the LMP management service shared library		
Function name	Referencing incoming LED-associated settings		
Symbol	Elib_LMP_GetLED_LinkMode		
Format	int Elib_LMP_GetLED_LinkMode (unsigned int ap_id);		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
Return value	Type	I/O	Explanation
Ret	int	O	Fixed pattern: ELIB_LMP_FIXPATTERN Melody-associated pattern: ELIB_LMP_LINKMELO Parameter error: ELIB_LMP_NG
Include file	srv_lmp.h		
Function explanation	<p>This function is used to reference the setting of the incoming LED-associated pattern ("fixed pattern" or "melody-associated pattern").</p> <p>Return value ret depends on the incoming LED-associated pattern specified. Either of the following values is used:</p> <p>ELIB_LMP_FIXPATTERN: Fixed pattern</p> <p>ELIB_LMP_LINKMELO: Melody-associated pattern</p>		
Related message	-		
Required procedure	-		
Notes	-		
Inhibitions	-		
Example of use	<pre>int ret; ret = Elib_LMP_GetLED_LinkMode ("Application ID");</pre>		

8.18 Registering incoming LED-associated Settings

8-18 Registering Incoming LED-associated Settings			
Classification	Function group provided by the LMP management service shared library		
Function name	Registering incoming LED-associated settings		
Symbol	Elib_LMP_SetLED_LinkMode		
Format	int Elib_LMP_SetLED_LinkMode (unsigned int ap_id , int set_val);		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
set_val	int	I	Setting value Specifies the setting value to be saved to the SRAM. (See the function explanation for details.)
Return value	Type	I/O	Explanation
Ret	int	O	Normal end: ELIB_LMP_OK Parameter error: ELIB_LMP_NG
Include file	srv_imp.h		
Function explanation	<p>This function is used to set and register the incoming LED-associated pattern ("fixed pattern" or "melody-associated pattern").</p> <p>Details of set_val (second argument): Specify either of the following values in set_val:</p> <p>ELIB_LMP_LED_FIXPATTERN: Fixed pattern</p> <p>ELIB_LMP_LED_LINKMELO: Melody-associated pattern</p>		
Related message	-		
Required procedure	-		
Notes	-		
Inhibitions	-		
Example of use	<pre>int ret; ret = Elib_LMP_SetLED_LinkMode ("Application ID", ELIB_LMP_LED_LINKMELO); /* Melody-associated pattern */</pre>		

8.19 Notifying of Change in Operation Modes

8-19 Notifying of Change in Operation Modes			
Classification	Function group provided by the LMP management service shared library		
Function name	Notifying of change in operation modes		
Symbol	Elib_LMP_ModeNotify		
Format	int Elib_LMP_ModeNotify (unsigned int ap_id , int mode);		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
Mode	int	I	Operation mode (See the function explanation for details.)
Return value	Type	I/O	Explanation
Ret	int	O	Normal end: ELIB_LMP_OK Abnormal end: ELIB_LMP_NG
Include file	srv_imp.h		
Function explanation	<p>This function is used to notify the LMP management service of change in operation modes.</p> <p>When an application notifies the LMP management service of change in operation modes, the service controls the backlight or incoming LED depending on each state.</p> <p>This function can be used for the LMP management service to control turning on the backlight or incoming LED.</p> <p>Details of mode (second argument) <Notification of melody selection screen display></p> <ul style="list-style-type: none"> - When the screen has shifted to the melody selection screen: ELIB_LMP_MELODY_SELECT - When the melody selection screen has shifted to another screen: ELIB_LMP_MELODY_FREE <p>(1) When the screen has shifted to the melody selection screen, an application must use this function to notify the LMP management service of the screen change (by specifying ELIB_LMP_MELODY_SELECT). This notification allows the LMP management service to control turning on the LED being recharged. For example, if the melody is switched with upward and downward key operation on the melody selection screen, the LMP management service prevents the LED being recharged from being turned on.</p> <p>(2) When the melody selection screen has shifted to another screen, an</p>		

	<p>application must use this function to notify the LMP management service of the screen change (by specifying <code>ELIB_LMP_MELODY_FREE</code>). This notification allows the LMP management service to normally control turning on the LED being recharged.</p> <p><Notification of the state in which shift to power-saving mode is possible>Shift is impossible.</p> <ul style="list-style-type: none"> - Notification of the state in which shift to power-saving mode is possible: <code>ELIB_LMP_SAVEMODE_POSSIBLE</code> - Notification of the state in which shift to power-saving mode is impossible: <code>ELIB_LMP_SAVEMODE_IMPOSSIBLE</code> <p>(1) When the state has shifted to that in which power-saving mode is operable, an application must use this function to notify the LMP management service of the state change (by specifying <code>ELIB_LMP_SAVEMODE_POSSIBLE</code>). This notification allows the LMP management service to control the backlight so that it will be turned off completely after the power-saving wait time has elapsed.</p> <p>(2) When the state has shifted to that in which power-saving mode is not operable, an application must use this function to notify the LMP management service of the state change (by specifying <code>ELIB_LMP_SAVEMODE_IMPOSSIBLE</code>). This notification allows the LMP management service to control the backlight so that it will be turned off incompletely (remain on faintly) after the power-saving wait time has elapsed.</p> <p><Notification of the state of TV phone communication></p> <ul style="list-style-type: none"> - Notification of starting TV phone communication: <code>ELIB_LMP_TVTALK_START</code> - Notification of ending TV phone communication: <code>ELIB_LMP_TVTALK_END</code> <p>(1) When the TV phone reception (sending) state has shifted to the state of TV phone communication, an application must use this function to notify the LMP management service of the state change (by specifying <code>ELIB_LMP_TVTALK_START</code>). This notification allows the LMP management service to perform competitive operation (key operation, recharger installation and removal, etc.) during backlight control. The illumination setting for TV phone communication (always on or on for 15 seconds) is used for competitive operation.</p> <p>(2) When the state of TV phone communication has ended, an application must use this function to notify the LMP management service of the end state (by specifying <code>ELIB_LMP_TVTALK_END</code>). This notification allows the LMP management service to control the backlight according to the illumination setting condition.</p> <p><Notification of using camera view mode></p> <ul style="list-style-type: none"> - Notification of starting camera view display: <code>ELIB_LMP_CAMERA_VIEW_START</code> - Notification of ending camera view display: <code>ELIB_LMP_CAMERA_VIEW_END</code>
--	---

	<p>(1) When the camera view screen is displayed, an application must use this function to notify the LMP management service of the state (by specifying ELIB_LMP_CAMERA_VIEW_START). This notification allows the LMP management service to control backlight competitive control (key operation, recharger installation and removal, etc.) during camera view mode display.</p> <p>(2) When camera view mode screen display has ended, an application must use this function to notify the LMP management service of the state (by specifying ELIB_LMP_CAMERA_VIEW_END). This notification allows the LMP management service to control return to the backlight lighting state according to the device state.</p> <p><Notification of stopping LED return control from recharge></p> <ul style="list-style-type: none"> - Notification of stopping LED return control from recharge: ELIB_LMP_CHARGE_CTRL_START - Notification of completion of stopping LED return control from recharge: ELIB_LMP_CHARGE_CTRL_END <p>(1) An application can skip turning the recharged LED on by the LMP management service during its LED control. To skip the processing, the application must use this function to notify the LMP management service of stopping LED return control from recharge. (To prevent the LMP management service from turning the LED on among a series of LED color lighting by the application.)</p> <p>This notification allows the application to exclusively control the LED so that the LMP management service does not perform LED return processing from recharge.</p> <p>(2) When LED control by an application is completed, the application must use this function to notify the LMP management service of completion of stopping LED return control from recharge.</p> <p>This notification allows the LMP management service to perform LED return control from recharge. The service performs LED return processing from recharge as well.</p>
Related message	-
Required procedure	-
Notes	-
Inhibitions	-
Example of use	<pre>int ret; /* When shifting to the melody selection screen */ ret = Elib_LMP_ModeNotify ("Application ID", ELIB_LMP_MELODY_SELECT);</pre>

8.20 Notifying of Change in Operation Modes

8-20 Referencing Communicating Illumination Settings			
Classification	Function group provided by the LMP management service shared library		
Function name	Referencing communicating illumination settings		
Symbol	Elib_LMP_GetTalkIllumination_Mode		
Format	Int Elib_LMP_GetTalkIllumination_Mode (unsigned int ap_id);		
Argument	Type	I/O	Explanation
ap_id	Unsigned int	I	Application ID
Return value	Type	I/O	Explanation
Ret	Int	O	<p>Normal end:</p> <p>Blinking pattern set to the SRAM. (See the function explanation for details.)</p> <p>Parameter error:</p> <p>ELIB_LMP_NG</p>
Include file	Srv_lmp.h		
Function explanation	<p>This function is used to reference the setting values (saved to the SRAM) of the emblem blinking patterns (colors) during communication.</p> <p>Details of the return value: Return value ret depends on the setting. One of the following return values is used:</p> <p>ELIB_LMP_ILLUM_COLOR1: Communicating color 1 (blue)</p> <p>ELIB_LMP_ILLUM_COLOR2: Communicating color 2 (color between blue and green)</p> <p>ELIB_LMP_ILLUM_COLOR3: Communicating color 3 (green)</p> <p>ELIB_LMP_ILLUM_COLOR4: Communicating color 4 (color between green and red)</p> <p>ELIB_LMP_ILLUM_COLOR5: Communicating color 5 (red)</p> <p>ELIB_LMP_ILLUM_COLOR6: Communicating color 6 (color between red and blue)</p> <p>ELIB_LMP_ILLUM_COLOR7: Communicating color 7 (white)</p> <p>ELIB_LMP_ILLUM_COLOR8: Gradation pattern 1</p> <p>ELIB_LMP_ILLUM_COLOR9: Gradation pattern 2</p> <p>ELIB_LMP_ILLUM_COLOR10: Gradation pattern 3</p> <p>ELIB_LMP_ILLUM_OFF: Communicating illumination off</p>		
Related message	-		
Required procedure	-		

Notes	-
Inhibitions	-
Example of use	<pre>Int ret; /* Reference the communicating emblem and LED lighting settings. */ Ret = Elib_LMP_GetTalkIllumination_Mode ("Application ID");</pre>

8.21 Registering Communicating Illumination Settings

8-21 Registering Communicating Illumination Settings			
Classification	Function group provided by the LMP management service shared library		
Function name	Registering communicating illumination settings		
Symbol	Elib_LMP_SetTalkIllumination_Mode		
Format	Int Elib_LMP_SetTalkIllumination_Mode (unsigned int ap_id , int set_val) ;		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
set_val	Int	I	Setting value Specifies the setting value to be saved to the SRAM. (See the function explanation for details.)
Return value	Type	I/O	Explanation
Ret	Int	O	Normal end: ELIB_LMP_OK Parameter error: ELIB_LMP_NG
Include file	Srv_Imp.h		
Function explanation	<p>This function is used to register the communicating emblem blinking pattern (color) in the SRAM.</p> <p>Details of set_val (second argument): Specify one of the following values in set_val:</p> <ul style="list-style-type: none"> ELIB_LMP_ILLUM_COLOR1: Communicating color 1 (blue) ELIB_LMP_ILLUM_COLOR2: Communicating color 2 (color between blue and green) ELIB_LMP_ILLUM_COLOR3: Communicating color 3 (green) ELIB_LMP_ILLUM_COLOR4: Communicating color 4 (color between green and red) ELIB_LMP_ILLUM_COLOR5: Communicating color 5 (red) ELIB_LMP_ILLUM_COLOR6: Communicating color 6 (color between red and blue) ELIB_LMP_ILLUM_COLOR7: Communicating color 7 (white) ELIB_LMP_ILLUM_COLOR8: Gradation pattern 1 ELIB_LMP_ILLUM_COLOR9: Gradation pattern 2 ELIB_LMP_ILLUM_COLOR10: Gradation pattern 3 ELIB_LMP_ILLUM_OFF: Communicating illumination off 		
Related message	-		
Required procedure	-		

Notes	-
Inhibitions	-
Example of use	<pre>Int ret; /* Register the communicating illumination blinking settings. */ ret = Elib_LMP_SetTalkIllumination_Mode ("Application ID", ELIB_LMP_ILLUM_COLOR1);</pre>

8.22 Registering Turn Rear Backlight On

8-22 Request to Turn Rear Backlight On			
Classification	Function group provided by the LMP management service shared library		
Function name	Request to turn rear backlight on		
Symbol	Elib_LMP_Backlight_BL_On		
Format	int Elib_LMP_Backlight_BL_On (unsigned int ap_id , int reason, ELIB_LMP_BL_CONFIGSET *bl_cfgset);		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
Reason	int	I	Specifies the reason for turning the rear backlight on. (See the function explanation for details.) Used to solve competitive operation.
bl_cfgset	_Elib_LMP_BL_CONFIGSET *	I	Rear backlight lighting attribute. (See the function explanation for details.) (1) Rear backlight lighting color (2) Rear backlight lighting color 2 (3) Rear backlight lighting mode (4) Rear backlight blinking mode (5) Rear backlight lighting cycle (6) Rear backlight off cycle (7) Lighting time (8) Lighting pattern
Return value	Type	I/O	Explanation
Ret	int	O	Normal end: ELIB_LMP_OK Cancelled to prioritize other requests: ELIB_LMP_CANCEL Parameter error: ELIB_LMP_NG
Include file	srv_lmp.h		
Function explanation	<p>This function is used to set the lighting attributes of the rear backlight and turn the rear backlight on.</p> <p>Details of reason (second argument) Specify this argument to determine how the rear backlight is turned on based on the current lighting status and reason for turning it on when the request is sent. Specify one of the following values in reason: (1) ELIB_LMP_AP_BL_IAPPLI; JAVA (releasing JavaAppli) (2) Set the corresponding reason immediately after displaying one of the following screens, and send a request to turn the rear backlight on: (The device open or close state may be switched due to open or close of the device and the rear LCD display may be changed. Also in this case, set the</p>		

corresponding reason immediately after displaying the screen, and send a request to turn the rear backlight on.)

[Reason for turning the rear backlight on] : [Screen display]

ELIB_LMP_BL_BATT_ARM : Low voltage alert

ELIB_LMP_BL_ALL_LOCK : All locked

ELIB_LMP_BL_POWERON : PIN1 input request, no USIM card
Incorrect USIM card, selecting subscriber

information

logo

ELIB_LMP_BL_CAMERA : Camera (when still images are
being shot), camera (when moving images are being shot)

ELIB_LMP_BL_TEL_RCV : Incoming

ELIB_LMP_BL_ANSWER_HOLD : Answer on hold

ELIB_LMP_BL_TEL_HOLD : Call on hold receiving

ELIB_LMP_BL_MEMO_START : Message slip starting

ELIB_LMP_BL_MEMO : Message slip recording

ELIB_LMP_BL_MAIL_RCVING: : Mail receiving

ELIB_LMP_BL_WAKE_ARM : Alarm clock function

ELIB_LMP_BL_SCH_ARM : Schedule alarm (including ToDo)

ELIB_LMP_BL_MAIL_RCVD : Mail reception completed

ELIB_LMP_BL_HOLD : On hold

ELIB_LMP_BL_TEL_CALL : Outgoing (calling)

ELIB_LMP_BL_HOOK : Communicating

ELIB_LMP_BL_64K : 64k data communicating or 64k data

incoming

ELIB_LMP_BL_PACKET : Packet communicating

ELIB_LMP_BL_PACKET_RCV : Packet incoming

ELIB_LMP_BL_USB : USB communicating

ELIB_LMP_BL_IR : Ir communicating, JavaAppli Ir

communicating

ELIB_LMP_BL_BKLTVOL : Contrast adjustment

ELIB_LMP_BL_MAIL_DSP : Contents of received mail

ELIB_LMP_BL_NEW_MAIL : Incoming or newly arriving mail,

newly arriving mail

ELIB_LMP_BL_ABSNBF : Missed incoming mail

ELIB_LMP_BL_ABSNBF_DSP : History of missed incoming mails

ELIB_LMP_BL_ABSNBF_DSP_NODATA : History of missed incoming

mails (30 or more incoming responses)

ELIB_LMP_BL_IAPPLI_PLAY_MELODY: JavaAppli melody playing

ELIB_LMP_BL_CLOCK : Date and time

ELIB_LMP_BL_IAPPLI_FREE : Bit map for releasing JavaAppli

ELIB_LMP_BL_REMORT_MONITORING : Remote monitoring

ELIB_LMP_BL_CAMERA_PREVIEW : Camera (at rear preview display)

ELIB_LMP_BL_CAMERA_CHECK : Camera (for checking images shot with
rear camera)


```

ELIB_LMP_BL_MSGR_RCVING      :      Message R receiving
ELIB_LMP_BL_MSGR_RCVD       :      Message R reception completed
ELIB_LMP_BL_MSGF_RCVING     :      Message F receiving message F
ELIB_LMP_BL_MSGF_RCVD      :      Message F reception completed
ELIB_LMP_BL_ADL_REWRITE     :      ADL software rewrite
ELIB_LMP_BL_CAMERA_PROCESS  :      Camera function processing

```

(start disabled, conversion, NG, and save)

(3) The rear LCD display may fail to be changed even after open or close state is switched due to open or close of the device (when the same display continues). In this case, set the following reason and send a request to turn the rear backlight on when receiving the open or close event.

[Reason for turning the rear backlight on] : [Screen display]

```
ELIB_LMP_BL_OPEN_CLOSE      : At open or close of the device
```

Details of bl_cfgset (third argument) lighting attribute

/* Structure for setting rear backlight lighting attributes */

```

typedef struct tag_Elib_LMP_BL_ConfigSet {
    unsigned short  Color1;          /* Rear backlight color specification 1 */
    unsigned short  Color2; /* Not used */
    unsigned short  Mode;            /* Rear backlight lighting mode */
    unsigned short  Blink_Mode;     /* Rear backlight blinking mode */
    unsigned short  Blink_Cycle;    /* Not used */
    unsigned short  Blink_Cycle2;   /* Not used */
    unsigned short  Time;           /* Rear backlight lighting time */
    unsigned short  Ptn;            /* Not used */
} _ELIB_LMP_BL_CONFIGSET ;

```

(1) Rear backlight lighting color: This Linux Platform supports the following value only:

ELIB_LMP_BL_COLOR1: Color 1 (blue/sapphire)

(2) Not used

(3) Rear backlight lighting mode:

ELIB_LMP_BL_ALWAYS: Always on

ELIB_LMP_BL_DURING: On for a given period of time

ELIB_LMP_BL_NOBAKLT: Off

ELIB_LMP_BL_NOCTR: Not controlled

(4) Rear backlight blinking mode

ELIB_LMP_BL_BLINK_OFF: Does not blink

(5) Not used

(6) Not used

(7) Lighting time:

Specify the rear backlight lighting time in seconds using a value from 1 to 100.

	<p>(This value is valid only when ELIB_LMP_BL_DURING (on for a given period of time) is specified in the lighting mode.)</p> <p>(8) Not used</p>
Related message	-
Required procedure	-
Notes	<p>(1) To turn the rear backlight off after turning it on using this function, perform Elib_LMP_Backlight_BL_Off (Request to Turn Rear Backlight Off).</p> <p>(2) When a request to turn the rear backlight on is sent for the rear backlight which is already on, the LMP management service determines how it will be turned on. Determination is based on the current lighting status and reason for turning it on when the request is sent.</p> <p>(3) If the rear backlight is controlled using the function that has priority over this request to turn the rear backlight on, the LMP management service cancels the request and returns return value ELIB_LMP_CANCEL.</p> <p>(4) When ELIB_LMP_BL_BLINK_OFF (Does not blink) is specified in the blinking mode, the blinking cycle is invalid. (The setting is not reflected.)</p> <p>(5) When ELIB_LMP_BL_ALWAYS (Always on) is specified in the lighting mode, the lighting time is invalid. (The setting is not reflected.)</p> <p>(6) Values can be omitted for the members with (not used) in the lighting attribute structure (third argument). (The LMP management service ignores the values specified in these members.)</p> <p>(7) If JAVA sends a request to control turning the rear backlight on or off due to releasing JavaAppli, the request can be received only when JAVA has obtained the rear backlight control authority. To obtain the authority, JAVA must use the JAVA backlight control management function (Elib_LMP_SetJavaCtrlMode). When JAVA does not have the control authority, the LMP management service returns ELIB_LMP_CANCEL.</p> <p>(8) In drive mode, to turn on the (alarm clock function) or (date and time) when the power is automatically turned on, specify ELIB_LMP_BL_NOBAKLT (Off) in the rear backlight lighting mode.</p> <p>(9) When the rear backlight control (on or off) need not be performed (e.g., to suppress flickering), specify ELIB_LMP_BL_NOCTR (Not controlled) in the rear backlight lighting mode.</p> <p>Example: Flickering in N504iS or N251i</p> <ul style="list-style-type: none"> - While the device is being closed, incoming mails (if any) are displayed when pressing the home key. <p>When the home key is repressed, the standby screen is redisplayed. Immediately after the standby clock is displayed, the backlight comes on. (Specifying ELIB_LMP_BL_NOCTR in the clock display can avoid flickering.)</p> <ul style="list-style-type: none"> - While the device is being closed, a display from JAVA on the rear side may be removed by pressing the side key. While holding down the side key, the backlight goes out and the display from JAVA is removed. When releasing the side key, the backlight comes on. (Specifying ELIB_LMP_BL_NOCTR in the clock display can avoid flickering.) <p>(10) For TV phone-related items (other than remote monitoring), use a normal</p>

	<p>voice reason and send a request to turn the rear backlight on.</p> <p>Example: During TV phone incoming, use ELIB_LMP_BL_TEL_RCV.</p> <p>(11) To display the following items on the rear LCD using the camera function, specify ELIB_LMP_BL_CAMERA_PROCESS (camera function processing (start disabled, conversion, NG, and save) in the reason and send a request to turn the rear backlight on:</p> <ul style="list-style-type: none"> - Camera start disabled - Converting - Shooting failed - Saving - Saving completed
Inhibitions	-
Example of use	<pre> int ret; /* Function return value */ _Elib_LMP_BL_CONFIGSET bl_cfgset; /* Lighting attribute */ bl_cfgset.Color = ELIB_LMP_BL_COLOR1; /* Specify color 1 for the rear backlight color. */ bl_cfgset.Mode = ELIB_LMP_BL_ALWAYS; /* Always on */ bl_cfgset.Blink_Mode = ELIB_LMP_BL_BLINK_OFF; /* Rear backlight on */ bl_cfgset.Time = 10; /* On for 10 seconds */ ret = Elib_LMP_Backlight_BL_On (i("Application ID", /* The requesting application has a one-shot date and time schedule. */ ELIB_LMP_AP_BL_IAPPLI, /* Specify pseudo-communication for the reason for turning the rear backlight on. */ &bl_cfgset); /* The rear backlight blinks over a 0.5-second cycle. */ </pre>

8.23 Registering Turn Rear Backlight Off

8-23 Request to Turn Rear Backlight Off			
Classification	Function group provided by the LMP management service shared library		
Function name	Request to turn rear backlight off		
Symbol	Elib_LMP_Backlight_BL_Off		
Format	int Elib_LMP_Backlight_BL_Off(unsigned int ap_id , int reason);		
Argument	Type	I/O	Explanation
ap_id	unsigned int	I	Application ID
reason	int	I	Reason for the request to turn the rear backlight off Specifies the reason for turning the rear backlight off. (See the function explanation for details.)
Return value	Type	I/O	Explanation
ret	int	O	Normal end: ELIB_LMP_OK Cancelled to prioritize other requests: ELIB_LMP_CANCEL Parameter error: ELIB_LMP_NG
Include file	srv_lmp.h		
Function explanation	<p>This function is used to turn the rear backlight off.</p> <p>Details of reason (second argument)</p> <p>Specify this argument to determine how the rear backlight is turned off based on the current lighting status and reason for turning it off when the request is sent.</p> <p>Specify either of the following values in reason:</p> <p>ELIB_LMP_AP_BL_IAPPLI: JAVA (releasing JavaAppli)</p> <p>ELIB_LMP_BL_SLEEP: Sleep setting</p>		
Related message	-		
Required procedure	-		
Notes	(1) When a request to turn the rear backlight off is sent after the rear backlight has been turned off, the LMP management service cancels the request and returns		

	<p>return value ELIB_LMP_CANCEL.</p> <p>(2) If JAVA sends a request to control turning the rear backlight on or off due to releasing JavaAppli, the request can be received only when JAVA has obtained the rear backlight control authority. To obtain the authority, JAVA must use the JAVA backlight control management function (Elib_LMP_SetJavaCtrlMode). When JAVA does not have the control authority, the LMP management service returns ELIB_LMP_CANCEL.</p> <p>(3) Applications other than JAVA can send a request to turn the rear backlight off only when sleep state enters after the display has disappeared. To send the request, ELIB_LMP_BL_SLEEP (sleep setting) must be specified in the reason for turning the rear backlight off.</p>
Inhibitions	-
Example of use	<pre> int ret; /* Function return value */ ret = Elib_LMP_Backlight_BL_Off ("Application ID", /* The requesting application has a one-shot date and time schedule. */ ELIB_LMP_AP_BL_IAPPLI); /* Specify pseudo-communication for the reason for turning the rear backlight off. */ </pre>

9. Sound System

9.1 Sound System Event Occurrence Notification Request

Classification	Functions for external processes of the sound system service		
Function	Sound system event occurrence notification request	Symbol	Elib_SS_Request
Functional overview	<p>Purpose: Requests notification of events that are posted by the sound system.</p> <ul style="list-style-type: none"> - Upon receiving the event occurrence message, the event to be posted to the application is registered. - Event occurrence can be known by executing the specified callback function. 		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_Request(unsigned int Ap_ID, int event, MsbFunc CallBack_ADDR);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
event	int	I	Type of event to be generated For a detailed description of the events, see 3.1 Event Formats.
CallBack_ADDR	MsbFunc	I	Callback function address
Return value	Type		
Ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>*For a use example, see Appendix 3.1 Event Formats.</p>		

9.2 Sound System Event Occurrence Notification Release

Classification	Functions for external processes of the sound system service		
Function	Sound system event occurrence notification release	Symbol	Elib_SS_Cancel
Functional overview	<p>Purpose: Releases notification requests of events that are posted by the sound system.</p> <p>- Releases events that are posted to the application.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_Cancel(unsigned int Ap_ID,int event);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
event	int	I	Type of registered event to be released For a detailed description of the events, see 3.1 Event Formats.
Return value	Type	-	
Ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	.		

9.3 Original Ring Tone Usage Status Collection

Classification	Functions for external processes of the sound system service		
Function	Original ring tone usage status collection	Symbol	Elib_SS_Get_UseOrgSound
Functional overview	<p>Purpose: Provides the high-level process with the status of using the registered original melodies for the various ring tones.</p> <p>- Accesses the usage status of the specified original melody.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_Get_UseOrgSound (unsigned int Ap_ID, _ELIB_SS_SOUNDINFOFLG_INFO *SoundInfoFlg);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
SoundInfoFlg	_ELIB_SS_SOUNDINFOFLG_INFO *	I/O	Pointer to the additional information structure
Return value	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>The additional information structure of the second argument is as follows.</p> <pre>typedef struct tag_ELIB_SS_SOUNDINFOFLG_INFO { unsigned int mode ; /* [I/-] Sound type */ unsigned short SoundID ; /* [I/-] Sound ID */ /* To view the usage status of all folders, */ /* specify ELIB_SS_S_ID_ALL. */ unsigned int link ; /* [-/O] Link destination information */ } _ELIB_SS_SOUNDINFOFLG_INFO ;</pre> <p>Mode corresponding to each bit of the argument link destination information</p> <ul style="list-style-type: none"> Bit 1: Ring tone selection Bit 2: Reserved bit Bit 3: i-mode mail Bit 4: Message free Bit 5: Message request Bit 6: Schedule alarm Bit 7: Reserved bit (fixed to 0) Bit 8: Wakeup function Bit 9: Telephone directory handy function (*1) Bit 10: Reserved bit (fixed to 0) Button confirmation sound 		

Bit 11: Reserved bit (fixed to 0) Dopa ring tone
Bit 12: Reserved bit (fixed to 0) Dopa voice ring tone
Bit 13: Voice/videophone ring tone (telephone directory specification)
Bit 14: Mail (telephone directory specification)
Bit 15: Voice/videophone ring tone (group specification)
Bit 16: Mail (group specification)
Bit 17: Non-notification setting ring tone
Bit 18: Public telephone ring tone
Bit 19: Notification disable ring tone
Bit 20: Videophone ring tone
Bit 21: ToDo alarm sound
Bit 22: Program edit setting sound
Bit 2: Random melody sound
Bit 24: Unused
Bit 25: Unused
Bit 26: Unused
Bit 27: Unused
Bit 28: Unused
Subsequent bits: Reserved

*1: If the telephone directory handy function is being used in the telephone directory specification or group specification, in use will occur (when any of the bits 13 to 16 are being used).

*2: If the specified sound ID is not being used by any function, the decision must be made using ELIB_SS_NOT_USE.

9.4 Original Ring Tone Unused Area Information Collection

Classification	Functions for external processes of the sound system service		
Function	Original ring tone unused area information collection	Symbol	Elib_SS_Get_FreeSpace
Functional overview	<p>Purpose: Collects the original ring tone unused area information.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_Get_FreeSpace(unsigned int Ap_ID);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Return value	Type	-	
FreeSpace	int	O	Stores the original ring tone unused area size Unit: Kilobytes Parameter error: ELIB_SS_PRMERR
Remark	<p>Use example:</p> <pre> unsigned int Ap_ID; /* Application ID */ int FreeSpace; /* Original ring tone unused area */ /* Application name */ FreeSpace = Elib_SS_Get_FreeSpace(Ap_ID); </pre>		

9.5 Original Ring Tone Unused Area Information Collection

Classification	Functions for external processes of the sound system service		
Function	Direct melody information collection	Symbol	Elib_SS_Melody_GetMelody_Info
Functional overview	<p>Purpose: Distributes the detailed information of the melody data to the high-level processes.</p> <p>Analyzes the melody data passed from the calling side and returns the detailed information as the result.</p>		
Include file	srv_ss.h		
Calling sequence	<pre>int Elib_SS_Melody_GetMelody_Info(unsigned int Ap_ID, _ELIB_SS_FGMELO *para, _ELIB_SS_ORG_DATA_INFO *Get_SoundData_Info);</pre>		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Para	_ELIB_SS_FGMELO *	I	Pointer to the melody data temporary storage structure
Get_SoundData_Info	_ELIB_SS_ORG_DATA_INFO *	O	Pointer to the direct melody information collection structure
Return value	Type		
Ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMEERR
Remark	<p>Use example:</p> <pre>unsigned int Ap_ID; /* Application ID */ _ELIB_SS_FGMELO para; /* Data temporary storage area structure */ _ELIB_SS_ORG_DATA_INFO Get_SoundData_Info; /* Original ring tone data collection structure */ int ret; /* Function return value */ /* Initialization */ memset(&para, 0x00, sizeof(_ELIB_SS_FGMELO)); memset(&Get_SoundData_Info, 0x00, sizeof(_ELIB_SS_ORG_DATA_INFO)); Ap_ID = 1; /* Parameter setting */ para.format = ELIB_SS_SMF; para.Sound_Len = len; para.Sound_Data = data; ret = Elib_SS_Melody_GetMelody_Info(Ap_ID, &para, &Get_SoundData_Info);</pre>		

9.6 Original Ring Tone Information Modification

Classification	Functions for external processes of the sound system service		
Function	Original ring tone information modification	Symbol	Elib_SS_Set_SoundData_Info
Functional overview	<p>Purpose: Modifies the detailed information of the melody data registered for the original ring tone.</p> <p>*This function updates all of the items that are targets of modification. For those items that need not be modified, set so that the current registration contents are set. (Use original ring tone collection and so on to collect and set the detailed information.)</p> <p>*This function must be called only when the melody data is updated.</p>		
Include file	srv_ss.h		
Calling sequence	<pre>int Elib_SS_Set_SoundData_Info(unsigned int Ap_ID, _ELIB_SS_ORGMELO_SET_PARAM *Set_SoundParam, _ELIB_SS_ORG_DATA_INFO *Set_SoundData_Info);</pre>		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Set_SoundParam	_ELIB_SS_ORGMELO_SET_PARAM *	I	Pointer to the original ring tone setting parameter structure
Set_SoundData_Info	_ELIB_SS_ORG_DATA_INFO *	I	Pointer to the original ring tone information modification structure
Return value	Type		
Ret	Int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>The second argument original ring tone setting parameter structure is as follows.</p> <pre>typedef struct tag_ELIB_SS_ORGMELO_SET_PARAM { unsigned int mode ; /* [I/-] Sound type */ int category ; /* [-/-] Folder type */ /*(Unused) unsigned short SoundID ; /* [I/-] Sound ID */ char dummy[2] ; /* [-/-] Boundary */ } _ELIB_SS_ORGMELO_SET_PARAM ;</pre> <p>The third argument original ring tone information modification structure is as follows.</p>		

```

typedef struct tag_ELIB_SS_ORG_DATA_INFO {
    int    format                /* [-/-] Data format          */Unused
    int    cp_mode;              /* [-/-] Procurement path identifier */Unused
    int    sorc;                 /* [I/-] Copyright (redistribution not permitted) information */
    int    edit;                 /* [-/-] Editing enabled/disabled information */Unused
    int    led;                  /* [-/-] Ring tone illumination linkage information */Unused
    int    vib;                  /* [-/-] Vibration linkage information */Unused
    unsigned long    file_size;    /* [-/-] File size          */Unused
    unsigned long    Sound_Len;    /* [-/-] Melody size        */Unused
    char version[ELIB_SS_MELO_VER_MAX]; /* [-/-] Version information */Unused
    char user_sound_title[ELIB_SS_MELO_TITLE_MAX+1]; /* [I/-] User-edited title */
    char mfi_sound_title[ELIB_SS_MELO_TITLE_MAX+1];
                                /* [-/-] Melody internal save title */Unused
    char user_filename[ELIB_SS_FILENAME_MAX+1]; /* [I/-] User-edited file name */
    char mail_filename[ELIB_SS_FILENAME_MAX+1];
                                /* [-/-] Mail attachment file name */Unused
    char copyright[ELIB_SS_MELO_CPRT_MAX+1]; /* [-/-] Copyright information */Unused
    unsigned char uim_id[ELIB_SS_UIM_CODE_MAX];
                                /* [-/-] UIM Card number      */Unused
    char dummy[1] ;              /* Boundary dummy            */Unused
    _ELIB_SS_ORGMELO_DATETIME datetime;
                                /* [-/-] Storage date and time */Unused

} _ELIB_SS_ORG_DATA_INFO;

```

Use example:

```

unsigned int Ap_ID;                /* Application ID          */
_ELIB_SS_ORGMELO_SET_PARAM Set_SoundParam ;
                                /* Original ring tone setting parameter structure */
_ELIB_SS_ORG_DATA_INFO Set_SoundData_Info ;
                                /* Original ring tone data modification structure */
int    ret ;                      /* Function return value   */

```

```
Ap_ID = 1;
```

```
/* Initialization */
```

```
memset( &Set_SoundParam, 0x00, sizeof _ELIB_SS_ORGMELO_SET_PARAM ) ;
```

```
memset( &Set_SoundData_Info, 0x00, sizeof _ELIB_SS_ORG_DATA_INFO ) ;
```

```
/* Collecting melody detailed information */
```

```
ret = Elib_SS_Get_SoundData_Info(Ap_ID, &Set_SoundParam, &Set_SoundData_Info ) ;
```

```
/* Modifying to redistribution not permitted */
```

```
Set_SoundData_Info.sorc = ELIB_SS_SORC_INFO_ON ;
```

```
/* Modifying the title name */
```

```
memset( &Set_SoundData_Info.user_sound_title[0], 0x00, (ELIB_SS_MELO_TITLE_MAX+1) ) ;
```

```
memcpy( &Set_SoundData_Info.user_sound_title[0], "USER_SOUNDTITLE", 15 );

/* Modifying the file name */
memset( &Set_SoundData_Info.user_filename[0], 0x00, (ELIB_SS_FILENAME_MAX+1) );
memcpy( &Set_SoundData_Info.user_filename[0], "USER_FILENAME.mid", 17 );

/* Modifying the information */
ret = Elib_SS_Set_SoundData_Info(Ap_ID, &Set_SoundParam, &Set_SoundData_Info );
```

9.7 Original Ring Tone Data Collection

Classification	Functions for external processes of the sound system service		
Function	Original ring tone data collection	Symbol	Elib_SS_Get_SoundData
Functional overview	<p>Purpose: Distributes the melody data registered for the original melody to the high-level processes.</p> <p>The following original ring tone data is collected.</p> <p>Original ring tone data</p> <ul style="list-style-type: none"> - Original ring tone data in MIDI format <p>If an unsupported data collection format is specified, an error (ELIB_SS_NG) will be returned for the return value.</p>		
Include file	srv_ss.h		
Calling sequence	<pre>int Elib_SS_Get_SoundData(unsigned int Ap_ID, _ELIB_SS_ORGMELO_SET_PARAM *Get_SoundID, _ELIB_SS_FGMELO *Get_SoundData);</pre>		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Get_SoundID	_ELIB_SS_ORGMELO_SET_PARAM *	I	Pointer to the original ring tone setting parameter structure
Get_SoundData	_ELIB_SS_FGMELO *	I/O	Pointer to the original ring tone data collection structure
Return value	Type	-	
Ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>The second argument original ring tone setting parameter structure is as follows.</p> <pre>typedef struct tag_ELIB_SS_ORGMELO_SET_PARAM { unsigned int mode ; /* [I/-]Sound type */ int category ; /* [-/-]Folder type */(Unused) unsigned short SoundID ; /* [I/-]Sound ID */ char dummy[2] ; /* [-/-]Boundary */ } _ELIB_SS_ORGMELO_SET_PARAM ;</pre> <p>Collecting data from the OMFN in test mode</p> <p>The following value must be specified when collecting data from the OMFN in test mode.</p> <p>:ELIB_SS_ORGMELO_TEST (for collecting data in test mode)</p>		

The third argument original ring tone data collection structure is as follows.

The calling side must allocate sufficient space for the pointer storage area in the structure.

```
typedef struct tag_ELIB_SS_FGMELO
{
    int          format ;           /* (I/-) Original ring tone collection format */
    unsigned long Sound_Len ;       /* (-/O) Data size (bytes)          */
    unsigned char * Sound_Data ;    /* (-/O) Data                      */
                                   /* The calling side must set the pointer to the storage area in advance. */
} _ELIB_SS_FGMELO ;
```


9.8 Original Ring Tone Information Collection

Classification	Functions for external processes of the sound system service		
Function	Original ring tone information collection	Sym bol	Elib_SS_Get_SoundData_Info
Functional overview	<p>Purpose: Distributes the detailed information of the melody data registered for the original melody to the high-level processes.</p> <p>The original ring tone data collection function must be used to collect the actual melody data.</p>		
Include file	srv_ss.h		
Calling sequence	<pre>int Elib_SS_Get_SoundData_Info(unsigned int Ap_ID, _ELIB_SS_ORGMELO_SET_PARAM *Get_SoundID, _ELIB_SS_ORG_DATA_INFO *Get_SoundData_Info);</pre>		
Argument	Type		Description
Ap_ID	unsigned int		Application ID
Get_SoundID	_ELIB_SS_ORGMELO_SET_PA RAM *		Pointer to the original ring tone setting parameter structure
Get_SoundData_Info	_ELIB_SS_ORG_DATA_INFO *		Pointer to the original ring tone information collection structure
Return value	Type		
Ret	Int		Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>The second argument original ring tone setting parameter structure is as follows.</p> <pre>typedef struct tag_ELIB_SS_ORGMELO_SET_PARAM { unsigned int mode ; /* [I/-]Sound type */ int category ; /* [-/]Folder type */(Unused) unsigned short SoundID ; /* [I/-]Sound ID */ char dummy[2] ; /* [-/]Boundary:1/4 */ } _ELIB_SS_ORGMELO_SET_PARAM ;</pre> <p>The fourth argument original ring tone information collection structure is as follows.</p> <pre>typedef struct tag_ELIB_SS_ORG_DATA_INFO {</pre>		

```

int      format                      /* OUT:  Data format      */
int      cp_mode;                    /* OUT:  Procurement path identifier */
int      sorc;                       /* OUT:  Copyright (redistribution not permitted) information */
int      edit;                       /* OUT: Editing enabled/disabled information */
int      led;                        /* OUT:  Ring tone illumination linkage */
int      vib;                        /* OUT:  Vibration linkage information */
unsigned long  file_size;             /* OUT:  File size          */
unsigned long  Sound_Len;             /* OUT:  Melody size        */
char          version[ELIB_SS_MELO_VER_MAX]; /* OUT:  Version information */
char          user_sound_title[ELIB_SS_MELO_TITLE_MAX+1]; /* OUT:  User-edited title */
char          mfi_sound_title[ELIB_SS_MELO_TITLE_MAX+1]; /* Unused */
char          user_filename[ELIB_SS_FILENAME_MAX+1]; /* OUT:  User-edited file name */
char          mail_filename[ELIB_SS_FILENAME_MAX+1]; /* OUT:  Mail attachment file name */
char          copyright[ELIB_SS_MELO_CPRT_MAX+1]; /* OUT:  Copyright information */
unsigned char  uim_id[ELIB_SS_UIM_CODE_MAX]; /* OUT:  UIM card number */
char          dummy[1];              /* Boundary dummy          */
_ELIB_SS_ORGMELO_DATETIME  datetime; /* OUT:  Date and time data (storage date and time) */
} _ELIB_SS_ORG_DATA_INFO ;

```

Collecting data from the OMFN in test mode

The following value must be specified for the sound type (mode) when collecting data from the OMFN in test mode.

:ELIB_SS_ORGMELO_TEST (for collecting data in test mode)

Use example:

```

unsigned int Ap_ID; /* Application ID */
_ELIB_SS_ORGMELO_SET_PARAM Get_SoundID; /* Original ring tone setting parameter structure */
_ELIB_SS_ORG_DATA_INFO Get_SoundData_Info; /* Original ring tone data collection structure */
int  ret; /* Function return value */

```

Ap_ID = 1;

```
ret = Elib_SS_Get_SoundData_Info(Ap_ID, &Get_SoundID, &Get_SoundData_Info);
```

9.9 Melody player list registration

Classification	Functions for external processes of the sound system service		
Function	Melody player list registration	Symbol	Elib_SS_MelodyPlayer_Set
Functional overview	<p>Purpose: Registers the sounds (INBOX, user-defined 1 to 20, and preinstalled melodies) to be played by the program player of the melody player.</p> <ul style="list-style-type: none"> - For the melody lists registered using this function, a sound title list and sound IDs can be collected using the sound title list and sound ID collection (Elib_SS_Get_Information) function. - When newly registering a melody player, ELIB_SS_PLAYER_RESET must be specified for the third argument mode and the melody player initialized. 		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_MelodyPlayer_Set (unsigned int Ap_ID, unsigned short PlayerID unsigned int mode, unsigned short SoundID);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
PlayerID	unsigned short	I	Player ID
mode	unsigned int	I	Sound type. For details, see Remark.
SoundID	unsigned short	I	Sound ID to be registered
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>Description of the second argument PlayerID</p> <p>Specify the sound ID to be collected when ELIB_SS_MELO_PLAYER is specified for the mode and ELIB_SS_CATEGORY_PLAYER for the category for sound title list and sound ID collection.</p> <p>The following can be specified for the third argument mode. For details, see E-5 Sound Title List and Sound ID Collection.</p> <p>ELIB_SS_MELO_PLAYER /* Sounds that can be used by the melody player */</p>		

ELIB_SS_PLAYER_RESET /* Resets the melody player.*/

To delete one melody of the program player, NULL must be specified for the third argument mode and the fourth argument SoundID.

Use example:

```
    unsigned int Ap_ID;           /* Application ID      */
    unsigned int  mode ;          /* Mode            */
    unsigned short SoundID ;      /* Sound ID */
    int          ret;             /* Function return value */
    unsigned short PlayerID ;     /*Player ID*/

    /*** Parameter setting ***/
    Ap_ID = 1;
    mode = ELIB_SS_MELO_PLAYER ;
    SoundID = ELIB_SS_S_ID_MELODY1 ;
    PlayerID = ELIB_SS_S_ID_MELOPLAY1 ;

    ret = Elib_SS_MelodyPlayer_Set( Ap_ID, PlayerID, mode, SoundID ) ;
```

9.10 Original Ring Tone Registration

Classification	Functions for external processes of the sound system service		
Function	Original ring tone registration	Symbol	Elib_SS_Set_Sound
Functional overview	<p>Purpose: Registers original melodies.</p> <p>- Melody data is registered or overwritten and registered. When a melody for which a ring tone has been set is overwritten, the current ring tone setting is retained.</p>		
Include file	srv_ss.h		
Calling sequence	<pre>int Elib_SS_Set_Sound(unsigned int Ap_ID, _ELIB_SS_ORGMELO_SET_PARAM *Set_param, _ELIB_SS_ORGMELO_SET *Set_sound);</pre>		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Set_param	_ELIB_SS_ORGMELO_SET_PARAM *	I/O	Pointer to the original ring tone setting parameter structure
Set_sound	_ELIB_SS_ORGMELO_SET *	I	Pointer to the original ring tone setting structure
Return value	Type	-	
ret	Int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR No free space: ELIB_SS_NOTSPACE
Remark			

Note

This function must not be called if the original melody data is not updated for the application. This function must be called only when the original melody data is updated. However, this restriction can be overlooked only when the update status for the application cannot be determined.

Use example:

```

unsigned int Ap_ID;                                /* Application ID */
_ELIB_SS_ORGMELO_SET_PARAM Set_param;             /* Original ring tone setting parameter structure */
_ELIB_SS_ORGMELO_SET Set_sound;                   /* Original ring tone setting structure */
_ELIB_SS_FGMELO Org_data;                         /* Original ring tone data structure */
int ret;                                           /* Function return value */
unsigned long melody_size;                        /* Format size */

/** Parameter setting */
Ap_ID = 1;
Set_param.mode = ELIB_SS_ORGMELO;
Set_param.category = ELIB_SS_CATEGORY_MELOINBOX;
Set_param.SoundID = ELIB_SS_ORGMELO_NEW;
Org_data.format = ELIB_SS_SMF;                    /* Format type */
Org_data.Sound_Len = melody_size;                 /* Data size (bytes) */
Org_data.Sound_Data = (unsigned char *)malloc( melody_size ); /* Data */
Set_sound.cp_mode = ELIB_SS_DOWNLOAD;
Set_sound.Sound_Addr = &Org_data;
ret = Elib_SS_Set_Sound(Ap_ID, &Set_param, &Set_sound);

```

9.11 Original Ring Tone Deletion

Classification	Functions for external processes of the sound system service		
Function	Original ring tone deletion	Symbol	Elib_SS_Del_Sound
Functional overview	<p>Purpose: Deletes the specified original ring tone. Multiple original ring tones can be specified.</p> <ul style="list-style-type: none"> - If the original ring tone to be deleted has been set as a ring tone, the setting will be restored to the default setting and the original ring tone deleted. - The event for which notification has been requested using the sound system event occurrence notification request (Elib_SS_Request) is posted to the high-level processes. The events that are posted by original ring tone deletion are as follows. <p>Original melody deletion completed notification (SoundNotify_DELSOUND)</p> <ul style="list-style-type: none"> - To release the notification request, use sound system event occurrence notification release (Elib_SS_Cancel). - When all of the melody data within a folder specified as random ring tones is deleted, the ring tone setting is returned to the default setting. <p>*1: See 1.2 Deleting Original Melodies Set for Various Ring Tones.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_Del_Sound(unsigned int Ap_ID, _ELIB_SS_DELMELO_ID *DelMelo_ID);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
DelMelo_ID	_ELIB_SS_DELMELO_ID *	I	Pointer to the deletion sound ID information structure
Return value	Type	-	
ret	Int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark			

Use example:

```

unsigned int Ap_ID;           /* Application ID      */
short del_id[5];             /* Deletion sound ID setting area */
_ELIB_SS_DELMELO_ID DelMelo_ID; /* Deletion sound ID list information structure */

/** Deleting sound IDs [1,2,5,7,10] */
del_id[0] = 1;
del_id[1] = 2;
del_id[2] = 5;
del_id[3] = 7;
del_id[4] = 10;

Ap_ID = 1;
/** Setting to the deletion list structure */
memset( &DelMelo_ID, 0x00, sizeof(_ELIB_SS_DELMELO_ID) );
DelMelo_ID.category = ELIB_SS_CATEGORY_MELOINBOX; /* INBOX */
DelMelo_ID.Delcnt = 5; /* Deletion count */
DelMelo_ID.SoundID = &del_id[0]; /* Deletion ID list */

ret = Elib_SS_Del_Sound(Ap_ID, &DelMelo_ID );

```


9.12 Melody Sort Setting

Classification	Functions for external processes of the sound system service		
Function	Melody sort setting	Symbol	Elib_SS_Set_Melody_Sort
Functional overview	<p>Purpose: Sets melody sorting.</p> <ul style="list-style-type: none"> - The contents set by this function are reflected to the following function. Sound title list and sound ID collection (Elib_SS_Get_Information) 		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_Set_Melody_Sort(unsigned int Ap_ID, _ELIB_SS_MELODYSORT_DATA *MelodySort_Data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
MelodySort_Data	_ELIB_SS_MELODYSORT_DATA *	I	Pointer to the melody sort data structure For details, see Remark.
Return value	Type	-	
ret	Int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>The second argument melody sort data structure is as follows.</p> <pre>typedef struct tag_ELIB_SS_MELODYSORT_DATA { int value ; /* [I-]: Sort key */ From newest: ELIB_SS_FROM_NEWEST From oldest: ELIB_SS_FROM_OLDEST Title ascending order: ELIB_SS_FROM_TITLE_ASC Title descending order: ELIB_SS_FROM_TITLE_DESC From biggest: ELIB_SS_FROM_BIG_SIZE From smallest: ELIB_SS_FROM_SMALL_SIZE File collection order: ELIB_SS_FROM_FILE_GET } _ELIB_SS_MELODYSORT_DATA ;</pre> <p>Use example:</p> <pre>unsigned int Ap_ID; /* Application ID */ _ELIB_SS_MELODYSORT_DATA MelodySort_Data ; /* Melody sort data structure */</pre>		

```
int    ret ;    /* Function return value    */

/** Parameter setting **/
Ap_ID = 1;
MelodySort_Data.value = Sort key ;

ret = Elib_SS_Set_Melody_Sort(Ap_ID, &MelodySort_Data ) ;
```

9.13 Melody Sort Access

Classification	Functions for external processes of the sound system service		
Function	Melody sort access	Symbol	Elib_SS_Get_Melody_Sort
Functional overview	<p>Purpose: Accesses melody sorting.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_Get_Melody_Sort(unsigned int Ap_ID, _ELIB_SS_MELODYSORT_DATA *MelodySort_Data);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
MelodySort_Data	_ELIB_SS_MELODYSORT_DATA *	O	Pointer to the melody sort data structure For details, see Remark.
Return value	Type	-	
ret	Int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>The third argument melody sort data structure is as follows.</p> <pre>typedef struct tag_ELIB_SS_MELODYSORT_DATA { int value ; /* [-O]: Sort key */ From newest: ELIB_SS_FROM_NEWEST From oldest: ELIB_SS_FROM_OLDEST Title ascending order: ELIB_SS_FROM_TITLE_ASC Title descending order: ELIB_SS_FROM_TITLE_DESC From biggest: ELIB_SS_FROM_BIG_SIZE From smallest: ELIB_SS_FROM_SMALL_SIZE File collection order: ELIB_SS_FROM_FILE_GET } _ELIB_SS_MELODYSORT_DATA ;</pre> <p>Use example:</p> <pre>unsigned int Ap_ID; /* Application ID */ _ELIB_SS_MELODYSORT_DATA MelodySort_Data ; /* Melody sort data structure */ int ret ; /* Function return value */</pre> <pre>Ap_ID = 1; ret = Elib_SS_Get_Melody_Sort(Ap_ID , &MelodySort_Data) ;</pre>		

9.14 User-Defined Folder List Collection

Classification	Functions for external processes of the sound system service		
Function	User-defined folder list collection	Symbol	Elib_SS_Get_FolderList
Functional overview	<p>Purpose: Distributes the list of user-defined folders to the high-level processes. The folder type (category) collected using this function can be used to collect sound titles. The sound ID list (Elib_SS_Get_Information) can be used to collect the sound title list for each folder.</p>		
Include file	srv_ss.h		
Calling sequence	<pre>int Elib_SS_Get_FolderList(unsigned int Ap_ID, _ELIB_SS_FOLDER_CATEGORYINFO *Folder_CategoryInfo, _ELIB_SS_FOLDER_CATEGORYLIST *Folder_CategoryList);</pre>		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Folder_CategoryInfo	_ELIB_SS_FOLDER_CATEGORYINFO *	I	Pointer to the folder collection specification structure
Folder_CategoryList	_ELIB_SS_FOLDER_CATEGORYLIST *	O	Pointer to the folder list data structure
Return value	Type	-	
ret	int	O	Normal end: Number of collected folders (0 to 20) Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>Description of the second argument _ELIB_SS_FOLDER_CATEGORYINFO folder type collection specification structure</p> <pre>typedef struct tag_ELIB_SS_FOLDER_CATEGORYINFO { int GetStartNumber; /* [I/-] Folder collection start number */ /* Specify the number from which collection is to be started for GetStartNumber. The number of folders specified for GetNumber is collected starting from GetStartNumber. The valid value is between 1 and total number of folders. */ int GetNumber; /* [I/-] Number of folders to be collected */ } _ELIB_SS_FOLDER_CATEGORYINFO;</pre> <p>Detailed description of the _ELIB_SS_FOLDER_CATEGORYINFO member GetNumber Number of titles to be collected <u>If 0 is specified</u>, folders will not be collected. <u>The total number of folders will be returned</u> as the return value.</p> <p>Description of the third argument _ELIB_SS_FOLDER_CATEGORYLIST folder list data structure</p> <pre>typedef struct tag_ELIB_SS_FOLDER_CATEGORYLIST { int category; /* [-/O] The folders corresponding to user-defined folders are stored. */ unsigned short TitleLen; /* [-/O] Determines whether folder information is present. */ }</pre>		

```

char TitleData[ELIB_SS_FOLNAME_MAX+1]; /* [-/O] The folder name is stored. */
char dummy[1]; /* [-/-] Boundary dummy */
}_ELIB_SS_FOLDER_CATEGORYLIST ;

```

*For the storage area, a buffer for the amount in the second argument

(structure _ELIB_SS_FOLDER_CATEGORYINFO member GetNumber) is required.

<Note>

- The following define values are provided as a means to check that the collected folder type (category) is within the parameter range.

Folder type minimum value (ELIB_SS_CATEGORY_MIN)

Folder type maximum value (ELIB_SS_CATEGORY_MAX)

Use example:

```

unsigned int Ap_ID; /* Application ID */
int cnt; /* Folder count collection variable */
int ret; /* Return value storage internal variable */
_ELIB_SS_FOLDER_CATEGORYINFO Folder_List_Info; /* Folder type collection specification structure */
_ELIB_SS_FOLDER_CATEGORYLIST Folder_CategoryList[20]; /* Folder type list data structure */

```

```
Ap_ID = 1;
```

```
/* Collecting the number of registered folders */
```

```
memset( &Folder_List_Info, 0x00, sizeof(_ELIB_SS_FOLDER_CATEGORYINFO) );
```

```
Folder_List_Info.GetStartNumber = 1 /* Folder collection start number */
```

```
Folder_List_Info.GetNumber = 0; /* Folder total count collection */
```

```
cnt = Elib_SS_Get_FolderList(Ap_ID, &Folder_List_Info, &Folder_CategoryList[0] );
```

```
/** Parameter setting */
```

```
memset( &Folder_List_Info, 0x00, sizeof(_ELIB_SS_FOLDER_CATEGORYINFO) );
```

```
memset( &Folder_CategoryList[0], 0x00, sizeof(Folder_CategoryList) );
```

```
Folder_List_Info.GetStartNumber = 1; /* Folder collection start number */
```

```
Folder_List_Info.GetNumber = cnt; /* Set the folder total count. */
```

```
ret = Elib_SS_Get_FolderList(Ap_ID, &Folder_List_Info, &Folder_CategoryList[0] );
```

9.15 Folder Melody Usage Status Collection

Classification	Functions for external processes of the sound system service		
Function	Folder melody usage status collection	Symbol	Elib_SS_Get_UseOrgSound_Folder
Functional overview	<p>Purpose: Distributes the usage status of the various ring tones of the melody data in the specified folder to the high-level processes.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_Get_UseOrgSound_Folder (unsigned int Ap_ID, _ELIB_SS_INFOFLG_INFO *InfoFlg_Info) ;		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
InfoFlg_Info	_ELIB_SS_INFOFLG_INFO *	I/O	Pointer to the additional information structure For details, see Remark.
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>The second argument additional information structure is as follows.</p> <pre>typedef struct tag_ELIB_SS_INFOFLG_INFO { unsigned int mode ; /* [-/-] Sound type */(Unused) int category ; /* [I/-] Folder type */ unsigned int link ; /* [-/O] Link destination information (*1) */ } _ELIB_SS_INFOFLG_INFO ;</pre> <p>*1:For details about the link destination information, see original ring tone usage status collection (Elib_SS_Get_UseOrgSound).</p> <p>Use example:</p> <pre>_ELIB_SS_INFOFLG_INFO InfoFlg_Info ; int ret ; /* Function return value */ /** Initialization **/ memset(&InfoFlg_Info, 0x00, sizeof(_ELIB_SS_INFOFLG_INFO)) ; /** Parameter setting **/ int Ap_ID; /* Application ID */</pre>		

```
InfoFlg_Info.category = ELIB_SS_CATEGORY_MELOUSER1 ;
```

```
ret = ELIB_SS_INFOFLG_INFO (Ap_ID, &InfoFlg_Info) ;
```

9.16 User-Defined Folder Creation Editing

Classification	Functions for external processes of the sound system service		
Function	User-defined folder creation editing	Symbol	Elib_SS_FolderCreate
Functional overview	<p>Purpose: Creates user-defined folders.</p> <ul style="list-style-type: none"> - To create a folder, new creation of each folder must be set for the argument member category. - To rename a folder, the folder type (category) of the target folder must be set for the argument member category. <p>* Only user-defined folders 1 to 20 can be created (otherwise, a parameter error will be returned).</p> <p>* The character string of the folder name is not checked for prohibited characters. The check must be made on the using side.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_FolderCreate(unsigned int Ap_ID, _ELIB_SS_FOLDERINFO * FolInfo);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
FolInfo	_ELIB_SS_FOLDERINFO *	I/O	Pointer to the folder information structure For details, see Remark.
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR Folder creation disabled: ELIB_SS_NOTCREATE
Remark	<p>The second argument folder information structure is as follows.</p> <pre>typedef struct tagELIB_SS_FOLDERINFO { int category ; /* [I/O] Folder type */ /* IN information */ /* <Creating a new user folder> */ /* ELIB_SS_FOLDER_CREATE: Creating a new user-defined folder */ /* <Renaming a user folder> */ /* ELIB_SS_CATEGORY_MELOUSER1: Renaming user-defined folder 1 */ /* */ /* ELIB_SS_CATEGORY_MELOUSER20: Renaming user-defined folder 20 */ /* OUT information: Valid only when creating a new user-defined folder is specified */ /* <When ELIB_SS_FOLDER_CREATE is specified> */ /* ELIB_SS_CATEGORY_MELOUSER1: User-defined folder 1 */ }</pre>		


```

/*      |      |      */
/*  ELIB_SS_CATEGORY_MELOUSER20:  User-defined folder 20      */
char  FolName[ELIB_SS_FOLNAME_MAX+1]; /* [!/-] Folder name      */
char  dummy[3];      /* Boundary dummy      */
} _ELIB_SS_FOLDERINFO ;

```

Description of the _ELIB_SS_FOLDERINFO member category

Specify the folder type.

Specify the folder type (category) collected using user-defined folder list collection (Elib_SS_Get_FolderList).

Use example:

```

unsigned int Ap_ID;      /* Application ID      */
_ELIB_SS_FOLDERINFO FolInfo ; /* Folder information structure */
int  ret ;      /* Function return value      */

/** Initialization **/
memset( &FolInfo, 0x00, sizeof(_ELIB_SS_FOLDERINFO) ) ;
Ap_ID = 1;

/** Parameter setting **/
FolInfo.category = ELIB_SS_FOLDER_CREATE ; /* Creating a new folder */
memcpy(FolInfo.FolName, "Alphabetical order", 10 ) ;

ret = Elib_SS_FolderCreate (Ap_ID, &FolInfo ) ;

```

9.17 User-Defined Folder Deletion

Classification	Functions for external processes of the sound system service		
Function	User-defined folder deletion	Symbol	Elib_SS_FolderErase
Functional overview	<p>Purpose: Deletes user-defined folders.</p> <p>- The sound data is also deleted even if sound data has been registered in the folder specified to be deleted.</p> <p>* Only user-defined folders 1 to 20 can be deleted (otherwise, a parameter error will be returned).</p> <p>* To collect the registration count of the sound data registered in the folder, the sound title list and sound ID collection (Elib_SS_Get_Information) interface must be used.</p> <p>* To collect the usage status of the ring tones of the sound data registered in the folder, the folder melody usage status collection (Elib_SS_Get_UseOrgSound_Folder) interface must be used.</p> <p>* If an original ring tone in a folder to be deleted has been set as a ring tone or a folder to be deleted has been set for a random melody, the ring tone setting will be restored to the default setting and the folder and all of the melodies within the folder deleted.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_FolderErase (unsigned int Ap_ID, _ELIB_SS_FOLDERINFO * FolInfo) ;		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
FolInfo	_ELIB_SS_FOLDERINFO *	I/O	Pointer to the folder information structure For details, see Remark.
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR Folder not registered: ELIB_SS_NOTFOLDER
Remark	<p>The second argument folder information structure is as follows.</p> <pre>typedef struct tagELIB_SS_FOLDERINFO { int category ; /* [I/-] Folder type */ /* <User-defined folder> */ /* ELIB_SS_CATEGORY_MELOUSER1: User-defined folder 1 */ /* */ /* ELIB_SS_CATEGORY_MELOUSER20: User-defined folder 20 */ char FolName[ELIB_SS_FOLNAME_MAX+1]; /* [-/-] Folder name */(Unused) char dummy[3] ; /* Boundary dummy */ } _ELIB_SS_FOLDERINFO ;</pre>		

Description of the _ELIB_SS_FOLDERINFO member category

Specify the folder type.

Specify the folder type (category) collected using user-defined folder list collection (Elib_SS_Get_FolderList).

Use example:

```
unsigned int Ap_ID;                /* Application ID */
_ELIB_SS_FOLDERINFO FolInfo ; /* Folder information structure */
int ret ;                        /* Function return value */

/** Initialization */
memset( &FolInfo, 0x00, sizeof(_ELIB_SS_FOLDERINFO) ) ;
Ap_ID = 1;
/** Parameter setting */
FolInfo.category = ELIB_SS_CATEGORY_MELOUSER1 ; /* Deletion folder setting */

ret = Elib_SS_FolderErase (Ap_ID, & FolInfo ) ;
```

9.18 Melody Data Folder Contents Move

Classification	Functions for external processes of the sound system service		
Function	Melody data folder contents move	Symbol	Elib_SS_FolderMove
Functional overview	<p>Purpose: Moves the melody data registered in the INBOX/user-defined folders 1 to 20 to the specified folder.</p> <p>Melody data can be moved only between the INBOX and user-defined folders 1 to 20 (otherwise, a parameter error will be returned).</p> <p>If all of the melody data within a folder specified as a random ring tone is moved, the ring tone setting will be restored to the default setting.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_FolderMove(unsigned int Ap_ID, _ELIB_SS_FOLDERMOVE * Folder_Move);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Folder_Move	_ELIB_SS_FOLDERMOVE *	I	Pointer to the folder move information structure
Return value	Type	-	
ret	Int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR Folder not registered: ELIB_SS_NOTFOLDER
Remark	<p>The second argument folder move information structure is as follows.</p> <pre>typedef struct tagELIB_SS_FOLDERMOVE { int num ; /* [I/-] Number of melody data items to be moved */ unsigned short *SoundID ; /* [I/-] Pointer to the array of melody data to be moved */ /* The sound IDs of the number (num) to be moved and */ /* the pointer must be specified. */ int category ; /* [I/-] Move destination folder number */ /* <Melody player folder> */ /* ELIB_SS_CATEGORY_MELOINBOX: INBOX folder */ /* ELIB_SS_CATEGORY_MELOUSER1: User-defined folder 1 */ /* */ /* ELIB_SS_CATEGORY_MELOUSER20: User-defined folder 20 */ } _ELIB_SS_FOLDERMOVE ;</pre>		

Description of the _ELIB_SS_FOLDERM MOVE member category

Specify the folder type.

Specify the folder type (category) collected using user-defined folder list collection (Elib_SS_Get_FolderList).

Use example:

```
unsigned int Ap_ID; /* Application ID */
_ELIB_SS_FOLDERM MOVE Folder_Move; /* Folder move information structure */
int ret; /* Function return value */
unsigned short SoundID; /* Move melody data sound ID */

/** Initialization */
memset( &Folder_Move, 0x00, sizeof(_ELIB_SS_FOLDERM MOVE) );
/** Parameter setting */
Ap_ID = 1;
SoundID = 1;
Folder_Move.num = 1;
Folder_Move.SoundID = &SoundID;
Folder_Move.category = ELIB_SS_CATEGORY_MELOUSER1;

ret = Elib_SS_FolderMove(Ap_ID, &Folder_Move );
```

9.19 Original Ring Tone Registration Folder Collection

Classification	Functions for external processes of the sound system service		
Function	Original ring tone registration collection	Symbol	Elib_SS_FolderNum
Functional overview	<p>Purpose: Collects the number of the folder for which the specified original ring tone is registered.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_FolderNum(unsigned int Ap_ID, unsigned short SoundID,int *category) ;		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
SoundID	unsigned short	I	Sound ID
category	int*	I/O	Folder type ELIB_SS_CATEGORY_MELOINBOX: INBOX folder ELIB_SS_CATEGORY_MELOUSER1 to 20: User-defined folder 1 to 20
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG No data: ELIB_SS_NODATA Parameter error: ELIB_SS_PRMERR
Remark	<p>Use example:</p> <pre> unsigned int Ap_ID ; /* Application ID */ int ret ; /* Function return value */ int category ; /* Folder type */ unsigned short SoundID ; /* Sound ID */ /*** Parameter setting ***/ Ap_ID = 1; /* Application ID */ SoundID = 1; ret = Elib_SS_FolderNum(Ap_ID, SoundID, & category) ; </pre>		

9.20 Ring Tone Volume Setting

Classification	Functions for external processes of the sound system service		
Function	Ring tone volume setting	Symbol	Elib_SS_RcvVolumeSet
Functional overview			
Purpose: Sets the ring tone volume.			
Include file	srv_ss.h		
Calling sequence	int Elib_SS_RcvVolumeSet(unsigned int Ap_ID, _ELIB_SS_SELECT_VOLUME *VolumeData);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
VolumeData	_ELIB_SS_SELECT_VOLUM E *	I	Pointer to the ring tone volume setting access structure For details, see Remark.
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark			

The second argument ring tone volume setting access structure is as follows.

```
typedef struct tag_ELIB_SS_SELECT_VOLUME { /* Sound system ring tone volume setting structure */
    unsigned int    mode; /* [I/-] Sound type See <Sound type> below. */
    int    SetData; /* [I/-] Setting data type See <Data type> below. */
    int    Volume; /* [I/-] The valid values of each ring tone volume can be collected as described in 3.4.6
                    Sound Volume Definition Values. When step tone (ELIB_SS_STEP) is
                    specified for StepTone for this function, the Volume value is automatically set to
                    the maximum value. */
    int    StepTone; /* [I/-] ELIB_SS_NOT_STEP: No step tone, ELIB_SS_STEP: Step tone*/
} _ELIB_SS_SELECT_VOLUME;
```

<Sound type>

Specify one of the following for the structure member mode.

```
ELIB_SS_ALLTELMELO /* Regular telephone ring tone */
ELIB_SS_MAIL /* Ring tone when receiving mail */
```

<Data type>

Specify one of the following for the structure member SetData.

```
ELIB_SS_SETALL /* Set ELIB_SS_SRAM and ELIB_SS_TONE. */
ELIB_SS_SRAM /* The user setting value is only written as nonvolatile data.
               The current ring tone volume status is not changed. */
ELIB_SS_TONE /* Only the current ring tone volume is set.
               The user setting value is not stored in the nonvolatile data. */
ELIB_SS_TEMP /* The ring tone is changed temporarily during playing.
               The volume is turned off temporarily until the next play. */
```


9.21 Ring Tone Volume Access

Classification	Functions for external processes of the sound system service		
Function	Ring tone volume access	Symbol	Elib_SS_RcvVolumeGet
Functional overview	<p>Purpose: Accesses the ring tone volume currently set.</p>		
Include file	srv_ss.h		
Calling sequence	<pre>int Elib_SS_RcvVolumeGet(unsigned int Ap_ID, _ELIB_SS_SELECT_VOLUME *VolumeData);</pre>		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
VolumeData	_ELIB_SS_SELECT_VOLUME *	I/O	Pointer to the ring tone volume setting access structure For details, see Remark.
Return value	Type	-	
ret	Int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark			

The second argument ring tone volume setting access structure is as follows.

```
typedef struct tag_ELIB_SS_SELECT_VOLUME { /* Sound system ring tone volume setting structure */
    unsigned int  mode; /* [I/-] Specify the sound type. See <Sound type> below. */
    int          SetData; /* [I/-] Specify the setting data type. See <Data type> below. */
    int          Volume; /* [-/O] The valid values of each ring tone volume can be collected as described in
                          3.4.6 Sound Volume Definition Values. When step tone (ELIB_SS_STEP) is
                          specified for StepTone for this function, the maximum value is stored for the
                          Volume value. */
    int  StepTone; /* [-/O] Step tone present/absent is stored.
                  ELIB_SS_NOT_STEP: No step tone
                  ELIB_SS_STEP: Step tone */
} _ELIB_SS_SELECT_VOLUME;
```

<Sound type>

Specify one of the following for the ring tone volume structure member mode.

```
ELIB_SS_ALLTELMELO /* Regular telephone ring tone */
ELIB_SS_MAIL       /* Ring tone when receiving mail */
```

<Data type>

Specify one of the following for the structure member SetData.

```
ELIB_SS_SRAM /* The nonvolatile data of the user setting value is accessed. */
ELIB_SS_TONE /* The current ring tone volume status is accessed. */
```

9.22 Talk Volume Setting

Classification	Functions for external processes of the sound system service		
Function	Talk volume setting	Symbol	Elib_SS_TalkVolumeSet
Functional overview	<p>Purpose: Sets the talk volume level.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_TalkVolumeSet(unsigned int Ap_ID, int Level);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Level	int	I	<p>Sets the talk volume level.</p> <p>The valid values of the talk volume can be collected as described in 3.4.6 Sound Volume Definition Values.</p>
Return value	Type	-	
ret	int	O	<p>Normal end: ELIB_SS_OK</p> <p>Abnormal end: ELIB_SS_NG</p> <p>Parameter error: ELIB_SS_PRMERR</p>
Remark			

9.23 Talk volume Access

Classification	Functions for external processes of the sound system service		
Function	Talk volume access	Symbol	Elib_SS_TalkVolumeGet
Functional overview	<p>Purpose: Accesses the setting value of the talk volume level.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_TalkVolumeGet(unsigned int Ap_ID, int * Level);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Level	int *	O	Pointer to the variable that stores the talk volume level The valid values of the talk volume can be collected as described in 3.4.6 Sound Volume Definition Values.
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark			

9.24 Built-in Hands Free On/Off Access

Classification	Functions for external processes of the sound system service		
Function	Built-in hands free on/off access	Symbol	Elib_SS_Get_OnOff_Handfree
Functional overview	<p>Purpose: Accesses the built-in hands free status (on/off).</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_Get_OnOff_Handfree(unsigned int Ap_ID, int *Get_param);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Get_param	int *	O	Built-in hands free setting status ELIB_SS_HANDFREE_ON: Built-in hands free on ELIB_SS_HANDFREE_OFF: Built-in hands free off
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>Use example:</p> <pre> unsigned int Ap_ID; /* Application ID */ int Get_param; /*Variable for accessing built-in hands free on/off */ int ret; /* Function return value */ /** Parameter setting */ Ap_ID = 1; ret = Elib_SS_Get_OnOff_Handfree(Ap_ID, &Get_param); </pre>		

9.25 Built-in Hands Free On/Off Setting

Classification	Functions for external processes of the sound system service		
Function	Built-in hands free on/off setting	Symbol	Elib_SS_Set_OnOff_Handfree
Functional overview	<p>Purpose: Sets the built-in hands free status (on/off).</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_Set_OnOff_Handfree(unsigned int Ap_ID, int Set_param);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Set_param	int	I	Built-in hands free setting value ELIB_SS_HANDFREE_ON: Built-in hands free on ELIB_SS_HANDFREE_OFF: Built-in hands free off
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMEERR
Remark	<p>Use example:</p> <pre> unsigned int Ap_ID; /* Application ID */ int Set_param; /* Variable for setting built-in hands free on/off */ int ret; /* Function return value */ /** Parameter setting */ Ap_ID = 1; Set_param = ELIB_SS_HANDFREE_ON; ret = Elib_SS_Set_OnOff_Handfree(Ap_ID, Set_param); </pre>		

9.26 Microphone Amplifier Gain Switching

Classification	Functions for external processes of the sound system service		
Function	Microphone amplifier gain switching	Symbol	Elib_SS_MicGainSet
Functional overview	<p>Purpose: Adjusts the telephone transmitter level when manner is pressed.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_MicGainSet (unsigned int Ap_ID, unsigned char Level) ;		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Level	unsigned char	I	Setting transmission level ELIB_SS_TON_UP: Increases the transmission level. ELIB_SS_TON_NORMAL: Sets the transmission level to normal.
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>This function must not be called if the calling side has not set the transmission path connection or the level setting is not changed.</p>		

9.27 Path Setting

Classification	Functions for external processes of the sound system service		
Function	Path setting	Symbol	Elib_SS_PathSet
Functional overview	<p>Purpose: Sets the path. Connects the telephone transmitter of the voice.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_PathSet (unsigned int Ap_ID, unsigned short Path)		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Path	unsigned short	I	Transmitting and receiving path connection setting ELIB_SS_PASMNG_TRANS: Transmitting path connection ELIB_SS_PASMNG_RECV: Receiving path connection ELIB_SS_PASMNG_TALK: Transmitting path and receiving path connection ELIB_SS_PASMNG_OFF: No connection
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>(1) If transmitting path connection is set while the transmitting and receiving paths are connected, the receiving path will be disconnected.</p> <p>(2) If receiving path connection is set while the transmitting and receiving paths are connected, the transmitting path will be disconnected.</p> <p>(3) Initial value: No connection</p> <p>(4) To transit to conversation, all sounds must be stopped by the high-level process and the conversation path set.</p> <p>(5) To transit to conversation, the mobile equipment operating status setting (SrvIF_SS_StatusSet) must be used to update the communication status in advance.</p> <p>(6) To disconnect, the mobile equipment operating status setting (SrvIF_SS_StatusSet) must be used after disconnecting to update the communication status.</p> <p>Note: The following definitions have the same values as the definitions in parentheses. However, taking uniformity of the interface into consideration, the definitions are scheduled to be deleted.</p> <p>ELIB_PASMNG_TRANS (ELIB_SS_PASMNG_TRANS): Transmitting path connection</p> <p>ELIB_PASMNG_RECV (ELIB_SS_PASMNG_RECV): Receiving path connection</p> <p>ELIB_PASMNG_TALK (ELIB_SS_PASMNG_TALK): Transmitting path and receiving path connection</p> <p>ELIB_PASMNG_OFF (ELIB_SS_PASMNG_OFF): No connection</p>		

9.28 Ring Tone Setting

Classification	Functions for external processes of the sound system service		
Function	Ring tone setting	Symbol	Elib_SS_Select_Sound
Functional overview	<p>Purpose: Sets the various ring tones.</p> <p>*However, this function does not set the specified ring tones or schedule ring tones and so on because the setting values are managed by the data exchange service and schedule service.</p> <ul style="list-style-type: none"> - Select the ring tone. - Use the sound ID to specify the ring tone. - The sound ID collected using Elib_SS_Get_Information must be specified for the sound ID. <p>*If video data is set for the ring tone, the ID collected by the movie control service must be specified.</p> <p>*If a random melody is specified for the ring tone, an INBOX (ELIB_SS_CATEGORY_MELOINBOX), user-defined folder 1 (ELIB_SS_CATEGORY_MELOUSER1) to 20 (ELIB_SS_CATEGORY_MELOUSER20), or preinstalled (ELIB_SS_CATEGORY_MELODEFAULT) folder must be specified as the ring tone.</p> <p>To play the ring tone, the folder registered as the ring tone must be decided and the appropriate sound ID used.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_Select_Sound (unsigned int Ap_ID, _ELIB_SS_SELECT_SOUND *Select_Sound);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Select_Sound	_ELIB_SS_SELECT_SOUND *	I	Pointer to the ring tone setting structure
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR Unsupported sound ID: ELIB_SS_NOSOUND
Remark	<p>The second argument ring tone setting structure is as follows.</p> <pre>typedef struct tag_ELIB_SS_SELECT_SOUND { /* Sound system ring tone setting structure */ unsigned int mode; /* [I/-] See Description below. */ unsigned short SoundID; /* [I/-] Sound ID */ char dummy[2]; /* [-/-] Boundary dummy (unused) */ } _ELIB_SS_SELECT_SOUND;</pre> <p>Detailed description of the structure member mode</p> <p>Specify one of the following.</p> <pre>ELIB_SS_ALLTELMELO /* All regular telephone ring tones*/ ELIB_SS_MAIL /* Ring tone when receiving mail */</pre>		

ELIB_SS_MESR	/* Ring tone when receiving a message request	*/
ELIB_SS_MESF	/* Ring tone when receiving message free	*/
ELIB_SS_184TELMELO	/* Non-notification setting ring tone	*/
ELIB_SS_PUBTELMELO	/* Public telephone ring tone	*/
ELIB_SS_IMPTTELMELO	/* Notification disabled ring tone	*/
ELIB_SS_VPMELO	/* Videophone ring tone	*/

9.29 Ring Tone Access

Classification	Functions for external processes of the sound system service		
Function	Ring tone access	Symbol	Elib_SS_Get_SoundTitle
Functional overview	<p>Purpose: Accesses the setting information of the various ring tones.</p> <p>* However, this function does not set the specified ring tones or schedule ring tones and so on because the setting values are managed by the data exchange service and schedule service.</p> <p>- Accesses the title and sound ID of the currently set sound ID that corresponds to the specified mode.</p> <p>*If video data has been set for the ring tone, only the video ID will be set. In addition, when play is requested, the sound ID below must be used to request play.</p> <p>- ELIB_SS_S_ID_IMOTION (video ring tone)</p> <p>*If a random melody has been set for the ring tone, only the folder ID is set.</p> <p>- INBOX (ELIB_SS_CATEGORY_MELOINBOX)</p> <p>-Use-defined folder 1 (ELIB_SS_CATEGORY_MELOUSER1) to 20 (ELIB_SS_CATEGORY_MELOUSER20)</p> <p>-Preinstalled (ELIB_SS_CATEGORY_MELODEFAULT)</p> <p>To play the ring tone, the folder registered as the ring tone must be decided and the appropriate sound ID used.</p> <p>*The methods below can be used to decide the type of sound ID that has been set.</p> <p>- video ID: If the result masked by 0x8000 is true.</p> <p>- Folder ID: If the result masked by ELIB_SS_S_ID_CATEGORY_MASK (0x1000) is true.</p> <p>- Sound ID: Other than the above</p>		
Include file	srv_ss.h		
Calling sequence	<pre>int Elib_SS_Get_SoundTitle(unsigned int Ap_ID, unsigned int mode, _ELIB_SS_SOUNDTITLE *Get_SoundTitle);</pre>		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
mode	unsigned int	I	See Remark.
Get_SoundTitle	_ELIB_SS_SOUNDTITLE *	O	Pointer to the ring tone setting information structure
Return value	Type	-	
ret	Int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark			

Detailed description of the second argument mode

Specify one of the following.

ELIB_SS_ALLTELMELO	/* All regular telephone ring tones */
ELIB_SS_MAIL	/* Ring tone when receiving mail */
ELIB_SS_MESR	/* Ring tone when receiving a message request */
ELIB_SS_MESF	/* Ring tone when receiving message free */
ELIB_SS_184TELMELO	/* Non-notification setting ring tone */
ELIB_SS_PUBTELMELO	/* Public telephone ring tone */
ELIB_SS_IMPTTELMELO	/* Notification disabled ring tone */
ELIB_SS_VPMELO	/* Videophone ring tone */

The third argument ring tone setting information structure is as follows.

```
typedef struct tag_ELIB_SS_SOUNDTITLE
{
    unsigned short SoundID ;      /* [-/O] The sound ID corresponding to the sound title is stored.*/
    unsigned short TitleLen ;     /* [-/O] Determines whether title data is present. */
    int category ;               /* [-/O] Folder type where the sound ID is stored.*/
    char title[ELIB_SS_MELO_TITLE_MAX+1] ; /* [-/O] The character string of the sound title is stored. */
    char dummy[1] ;              /* [-/] Boundary dummy (unused) */
} _ELIB_SS_SOUNDTITLE ;
```

Use example:

```
unsigned int Ap_ID;           /* Application ID */
unsigned int mode;            /* Sound type */
_ELIB_SS_SOUNDTITLE Get_SoundTitle; /* Original ring tone title access structure */
int ret;                      /* Function return value */

/** Parameter setting */
mode = ELIB_SS_ALLTELMELO;
/* Application ID */
Ap_ID = 1;
ret = Elib_SS_Get_SoundTitle(Ap_ID, mode, &Get_SoundTitle);
```

9.30 Ring Tone Sound Time Setting

Classification	Functions for external processes of the sound system service		
Function	Ring tone sound time setting	Symbol	Elib_SS_Set_Sound_Time
Functional overview	<p>Purpose: Stores the ring tone sound time to SRAM.</p> <p>- Sets the various ring tone sound times.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_Set_Sound_Time(unsigned int Ap_ID, unsigned int mode, int SoundTime);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
mode	unsigned int	I	Sound type
SoundTime	int	I	- Various ring tone sound times (seconds) The valid values of the various sound times can be collected as described in 3.4.1 Original Ring Tone Definitions.
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>Detailed description of the second argument mode</p> <p>For this function, the following values are fixed for the mode.</p> <p>ELIB_SS_MAIL /* Playing time of the M68 i-mode ring tones (including the SMS) */</p> <p>ELIB_SS_MESR /* Playing time of the M68 message request ring tones */</p> <p>ELIB_SS_MESF /* Playing time of the M68 message free ring tones */</p>		

9.31 Ring Tone Sound Time Access

Classification	Functions for external processes of the sound system service		
Function	Ring tone sound time access	Symbol	Elib_SS_Get_Sound_Time
Functional overview	<p>Purpose: Accesses the ring tone sound time.</p> <p>- Accesses the ring tone sound time.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_Get_Sound_Time(unsigned int Ap_ID, unsigned int mode, int *SoundTime);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
mode	unsigned int	I	Sound type
SoundTime	int *	O	- Various ring tone sound times (seconds). The valid values of the various sound times can be collected as described in 3.4.1 Original Ring Tone Definitions.
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>Detailed description of the second argument mode</p> <p>For this function, the following values are fixed for the mode.</p> <p>ELIB_SS_MAIL /* Playing time of the M68 i-mode ring tones (including the SMS) */</p> <p>ELIB_SS_MESR /* Playing time of the M68 message request ring tones */</p> <p>ELIB_SS_MESF /* Playing time of the M68 message free ring tones */</p>		

9.32 Key Sound On/Off Setting

Classification	Functions for external processes of the sound system service		
Function	Key sound on/off setting	Symbol	Elib_SS_OnOff_KeySound
Functional overview	<p>Purpose: Turns the key sound on/off.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_OnOff_KeySound(unsigned int Ap_ID, int mode);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
mode	int	I	ELIB_SS_KEY_ON: Key sound on ELIB_SS_KEY_OFF: Key sound off
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark			

9.33 Key Sound On/Off Access

Classification	Functions for external processes of the sound system service		
Function	Key sound on/off access	Symbol	Elib_SS_Get_OnOff_KeySound
Functional overview	<p>Purpose: Accesses the key sound on/off status.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_Get_OnOff_KeySound(unsigned int Ap_ID, int * mode);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
mode	int *	O	ELIB_SS_KEY_ON: Key sound on ELIB_SS_KEY_OFF: Key sound off
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark			

9.34 Default Sound ID Collection

Classification	Functions for external processes of the sound system service		
Function	Default sound ID collection	Symbol	Elib_SS_Get_Default_SoundID
Functional overview	<p>Purpose: Distributes the initial setting values of the various ring tones to the high-level processes.</p> <p>Collects the default sound ID of each mode.</p>		
Include file	srv_ss.h		
Calling sequence	<pre>int Elib_SS_Get_Default_SoundID(unsigned int Ap_ID, unsigned int mode, unsigned short *SoundID);</pre>		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
mode	unsigned int	I	Mode
SoundID	unsigned short *	O	Default sound ID
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Parameter error: ELIB_SS_PRMERR
Remark	<p>The following modes must be specified for the second argument.</p> <pre> ELIB_SS_ALARM /* Schedule alarm sound */ ELIB_SS_MADRESS /* Mail specified ring tone */ ELIB_SS_SITEIMELO /* Specified ring tone */ ELIB_SS_ALLTELMELO /* All regular telephone ring tones */ ELIB_SS_MAIL /* Ring tone when receiving mail */ ELIB_SS_MESR /* Ring tone when receiving a message request */ ELIB_SS_MESF /* Ring tone when receiving message free */ ELIB_SS_184TELMELO /* Non-notification setting ring tone */ ELIB_SS_PUBTELMELO /* Public telephone ring tone */ ELIB_SS_IMPTELMELO /* Notification disabled ring tone */ ELIB_SS_VPMELO /* Videophone ring tone */ ELIB_SS_TODO /* ToDo alarm sound */ </pre>		

9.35 Operation Mode Setting

Classification	Functions for external processes of the sound system service		
Function	Operation mode setting	Symbol	Elib_SS_ToneModeSet
Functional overview	<p>Purpose: Changes the path, volume, and gain settings that accompany modification of the operation mode. Modification is supported even if a tone or melody is being played.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_ToneModeSet(unsigned int Ap_ID, unsigned char mode);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
mode	unsigned char	I	AUDIO operation mode
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>The following AUDIO operation modes must be specified for the second argument.</p> <p>ELIB_SS_MODMNG_KEITAI: Mobile phone mode</p> <p>ELIB_SS_MODMNG_EARHON1: Earphone 1 mode</p> <p>ELIB_SS_MODMNG_EARHON2: Earphone 2 mode</p>		

9.36 DSP Power Supply Control

Classification	Functions for external processes of the sound system service		
Function	DSP power supply control	Symbol	Elib_SS_DspCtrl
Functional overview	<p>Purpose: Controls on/off of the DSP power supply based on the status specified for the mode.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_DspCtrl(unsigned int Ap_ID, unsigned char mode);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
mode	unsigned char	I	DSP control specification mode
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>The following DSP control modes must be specified for the second argument.</p> <p>ELIB_SS_DSPTALK_ON: AMR conversation start</p> <p>ELIB_SS_DSPTALK_OFF: AMR conversation stop</p> <p>ELIB_SS_DSPPLAY_ON: u-law playback start</p> <p>ELIB_SS_DSPPLAY_OFF: u-law playback stop</p> <p>ELIB_SS_DSPNOTEL_ON: No telephone on (unused)</p> <p>ELIB_SS_DSPNOTEL_OFF: No telephone off (unused)</p> <p>ELIB_SS_DSPMREC_ON: Movie record start</p> <p>ELIB_SS_DSPMREC_OFF: Movie record stop</p> <p>ELIB_SS_DSPMPLAY_ON: Movie playback start</p> <p>ELIB_SS_DSPMPLAY_OFF: Movie playback stop</p> <p>ELIB_SS_DSPAPLAY_ON: Movie AAC contents playback start</p> <p>ELIB_SS_DSPAPLAY_OFF: Movie AAC contents playback stop</p> <p>ELIB_SS_DSPTVTALK_ON: Videophone conversation start</p> <p>ELIB_SS_DSPTVTALK_OFF: Videophone conversation stop</p> <p>ELIB_SS_DSPREC_ON: Record start</p> <p>ELIB_SS_DSPREC_OFF: Record stop</p>		

9.37 Mute Control

Classification	Functions for external processes of the sound system service		
Function	Mute control	Symbol	Elib_SS_VolMute_Set
Functional overview	<p>Purpose: Controls muting of the volume.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_VolMute_Set(unsigned int Ap_ID, unsigned int mode, unsigned int mute);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
mode	unsigned int	I	Sound type
mute	unsigned int	I	Mute type ELIB_SS_ON: Set mute. ELIB_SS_OFF: Release mute. To set the mute function, specify ELIB_SS_ON. To release the mute function, specify ELIB_SS_OFF.
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>Notes:</p> <ul style="list-style-type: none"> - If mute is specified during mute, ELIB_SS_OK will be returned. - If mute is specified during silent, ELIB_SS_OK will be returned. - To stop, ELIB_SS_OFF must be set. 		

9.38 Movie Sound Record/Playback Status Setting

Classification	Functions for external processes of the sound system service		
Function	Movie sound record/playback status setting	Symbol	Elib_SS_MovieRecStsSet
Functional overview	<p>Purpose: Sets the movie sound record/playback status.</p> <p>Note: This function controls the paths based on the movie voice record/playback status.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_MovieRecStsSet(unsigned int Ap_ID, unsigned int MovieRecSts);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
MovieRecSts	unsigned int	I	Operating status ELIB_SS_MWAIT: Not operating ELIB_SS_MPLAY_AMR: AMR playback ELIB_SS_MREC_AMR: AMR record ELIB_SS_MPLAY_AAC8K: AAC playback (8 KHz) ELIB_SS_MPLAY_AAC16K: AAC playback (16 KHz)
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark			

9.39 DSP Parameter Modification

Classification	Functions for external processes of the sound system service		
Function	DSP parameter modification	Symbol	Elib_SS_DspParamChg
Functional overview	<p>Purpose: Modifies the headset information and audio I/O information for the DSP.</p> <p>Note: This function is used for the videophone.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_DspParamChg (unsigned int Ap_ID);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG
Remark			

9.40 Path Access

Classification	Functions for external processes of the sound system service		
Function	Path access	Symbol	Elib_SS_PathGet
Functional overview	<p>Purpose: Accesses the current path setting status.</p>		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_PathGet (unsigned int Ap_ID, unsigned short *Path)		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Path	unsigned short *	O	Transmitting and receiving path connection status ELIB_SS_PASMNG_TRANS: Transmitting path connection ELIB_SS_PASMNG_RECV: Receiving path connection ELIB_SS_PASMNG_TALK: Transmitting and receiving path connection ELIB_SS_PASMNG_OFF: No connection
Return value	Type	-	
ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark			

9.41 Media Sound Start

Classification	Functions for external processes of the sound system service		
Function	Media sound start	Symbol	Elib_SS_Play_MediaSound
Functional overview	<p>Purpose: Starts playing the melody data at the specified address.</p> <ul style="list-style-type: none"> - For details about repeating, see Remark. - When playing the sound, the vibrator conforms to the user setting. - The setting when the earphones are mounted conform to the earphone switch setting. - The events for which notification has been requested using the sound system event occurrence notification request (Elib_SS_Request) are posted to the high-level processes. The following events are posted when the media sound starts. <p>Melody sound start notification (SoundNotify_STARTSOUND_START) Melody sound stop notification (SoundNotify_STARTSOUND_END) Melody sound interrupt notification (SoundNotify_STOPSOUND)</p> <p>* For details about the events, see 1.3 Relationship Between Sounds and Events.</p> <p>To release the notification request, use sound system event occurrence notification release (Elib_SS_Cancel).</p> <ul style="list-style-type: none"> - The sound times are not controlled for the sound system. <p>Use media sound stop (Elib_SS_Stop_Sound) to request stopping or wait until the sound is completed playing. The sound times must be controlled using an application.</p> <p>* For details about the repeat sounds, see 1.3 Relationship Between Sounds and Events.</p> <ul style="list-style-type: none"> - The sound system does not release area that has been allocated for the sound data. Therefore, the area must be released by the process that allocated the area (calling side of this interface). The area must be released when processing returns from this interface. 		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_Play_MediaSound(unsigned int Ap_ID, _ELIB_SS_PLAY_MEDIASOUND *Play_MediaSound);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Play_MediaSound	_ELIB_SS_PLAY_MEDIASOUND *	I	Pointer to the media sound start information
Return value	Type	-	
ret	int	O	Play enabled: ELIB_SS_PLAY_OK Play disabled: ELIB_SS_PLAY_NG Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR

Remark

The media sound start information structure is as follows.

```
typedef struct tag_ELIB_SS_PLAY_MEDIASOUND {
    unsigned int  mode;           /* [I/-] See <mode setting type> below. */
    int           format;        /* [I/-] Specify the data format to be played.
                                   Format type:
    ELIB_SS_SMF (MIDI format)    unsigned int  Sound_Mode_Led;
                                   /* [I/-] LED color information for playing.  See <LED setting type> below.*/
    unsigned int  Option;        /* [I/-] Option information.  See <Option information> below. */
    unsigned char *Sound_Data;    /* [I/-] Sound data storage address */
    unsigned long Sound_Len;      /* [I/-] Sound data size */
} _ELIB_SS_PLAY_MEDIASOUND;
```

<mode setting type>

The <Sound type> and <Manner mode sound management> below must be specified for the second argument _ELIB_SS_PLAY_MEDIASOUND structure member mode. Specify by taking the logical add of <Sound type> and <Manner mode sound management> as shown below. See Use example for reference.

<Sound type>

```
ELIB_SS_MAIL      /* Ring tone when receiving mail */
ELIB_SS_MESR      /* Ring tone when receiving a message request */
ELIB_SS_MESF      /* Ring tone when receiving message free */
ELIB_SS_IMODE     /* Download melody sound test */
ELIB_SS_IAPPLI    /* i-application melody sound test */
ELIB_SS_QR        /* Two-dimensional bar code collection melody sound test */
ELIB_SS_SWF       /* SWF sound: Site effect sound */
```

mode = <Sound type> | <Manner mode sound management> ;

<Manner mode sound management>

```
ELIB_SS_SOUND_MANNER_NORMAL /*Based on the manner mode setting */
ELIB_SS_SOUND_MANNER_FORCE  /*When playing temporarily during manner mode*/
```

<LED setting type>

The <LED color specification> and <Multifunction key backlight> below must be specified for the second argument _ELIB_SS_PLAY_MEDIASOUND structure member Sound_Mode_Led. Specify by taking the logical add of <LED color specification> and <Multifunction key backlight> as shown below. See Use example for reference.

Sound_Mode_Led = <LED color specification> | <Multifunction key backlight>

<LED color specification>

```
ELIB_SS_LED_COLOR_OFF /* The LED is not turned on. */
ELIB_SS_LED_NO_SET    /* The LED is turned on as described below based on the sound type
specified for mode. */
ELIB_LMP_LED_COLOR1 to 13 /* Specify the illumination color defined using the ELIB LMP service. (*1)*/
                          /* The LED flashes using the illumination color set from the application. */
```

*1: This function is used when specified illumination or mail specified illumination has been set using the telephone

directory or specified handy function. When playing using ELIB_SS_MAIL, the specified illumination information accessed by the telephone directory application must be set for the LED color information (Sound_Mode_Led) if specified illumination has been set.

<Option information>

The option information must be set using the following bits (1 to 32).

Bit 1: Loop present/absent information (no loop: ELIB_SS_PLAY_LOOP_OFF, loop:

ELIB_SS_PLAY_LOOP_ON)

Bits 2 to 32: Reserved bits (Because these bits are reserved for future use, they must be initialized to 0.)

*** Key confirmation sound restrictions of the melody player**

During the sounds started using the sound types below, specific key confirmation sounds are not played because it is judged that the melody player is operating (specific key confirmation sounds: upper keys, lower keys, Clear key, and S2 key).

ELIB_SS_IMODE /* Download melody sound test */

ELIB_SS_IAPPLI /* i-application melody sound test */

ELIB_SS_QR /* Two-dimensional bar code collection melody sound test */

ELIB_SS_SWF /* SWF sound: Site effect sound */

*** Repeat sounds**

The repeat sounds are as follows.

<Repeated> ELIB_SS_MAIL

ELIB_SS_MESR

ELIB_SS_MESF

<Not repeated> ELIB_SS_IMODE

ELIB_SS_IAPPLI

ELIB_SS_QR

ELIB_SS_SWF

9.42 Media Sound Stop

Classification	Functions for external processes of the sound system service		
Function	Media sound stop	Symbol	Elib_SS_Stop_MediaSound
Functional overview	<p>Purpose: Stops playing the melody that was started by media sound start.</p> <p>- The events for which notification has been requested using the sound system event occurrence notification request (Elib_SS_Request) are posted to the high-level processes. The following event is posted when the media sound stops.</p> <p>Melody sound interrupt notification (SoundNotify_STOPSOUND)</p> <p>- To release the notification request, use sound system event occurrence notification release (Elib_SS_Cancel).</p> <p>* For details about the events, see 1.3 Relationship Between Sounds and Events.</p>		
Include file	srv_ss.h		
Calling sequence	Int Elib_SS_Stop_MediaSound(unsigned int Ap_ID, _ELIB_SS_STOP_MEDIASOUND *Stop_MediaSound);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Stop_MediaSound	_ELIB_SS_STOP_MEDIASOUND *	I	Pointer to the media sound stop information
Return value	Type	-	
ret	int	O	Normal: ELIB_SS_OK Abnormal: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>The media sound stop information structure is as follows.</p> <pre>typedef struct tag_ELIB_SS_STOP_MEDIASOUND { unsigned int mode; /* [I/-] See <mode setting type> below. */ int format; /* [I/-] Specify the data format to be played. Format type: ELIB_SS_SMF (MIDI format)*/ } _ELIB_SS_STOP_MEDIASOUND;</pre> <p><mode setting type></p> <p>Specify one of the following for the second argument _ELIB_SS_STOP_MEDIASOUND structure member mode.</p> <pre>ELIB_SS_MAIL /* Ring tone when receiving mail */</pre>		

ELIB_SS_MESR	/* Ring tone when receiving a message request */
ELIB_SS_MESF	/* Ring tone when receiving message free */
ELIB_SS_IMODE	/* Download melody sound test */
ELIB_SS_IAPPLI	/* i-application melody sound test */
ELIB_SS_QR	/* Two-dimensional bar code collection melody sound test */
ELIB_SS_SWF	/* SWF sound: Site effect sound */

9.43 Sound Start

Classification	Functions for external processes of the sound system service		
Function	Sound start	Symbol	Elib_SS_Play_Sound
Functional overview	<p>Purpose: Requests starting of the sound (ring tone, original melody, key sound, and so on).</p> <p>The following sound IDs can be specified.</p> <ul style="list-style-type: none"> * Sound IDs collected using Elib_SS_Get_Information. * Sound IDs accessed using Elib_SS_Get_SoundTitle. * Sound IDs set using schedule, specified handy function specified ring tone, and specified handy function mail specified ring tone. * For sound IDs other than the above, see the IDs defined for use as other sounds in Appendix 3.3 Sound IDs. <ul style="list-style-type: none"> - The events for which notification has been requested using the sound system event occurrence notification request (Elib_SS_Request) are posted to the high-level processes. The following events are posted when the sound starts. <ul style="list-style-type: none"> Alarm sound start notification (SoundNotify_ALARM_START) /*Alarm sound only*/ Alarm sound stop notification (SoundNotify_ALARM_END) /* Alarm sound only*/ Alarm sound interrupt notification (SoundNotify_ALARM_STOP) /* Alarm sound only*/ Melody sound start notification (SoundNotify_STARTSOUND_START) Melody sound stop notification (SoundNotify_STARTSOUND_END) Melody sound interrupt notification (SoundNotify_STOPSOUND) * For details about the events, see 1.3 Relationship Between Sounds and Events. - To release the notification request, use sound system event occurrence notification release (Elib_SS_Cancel). - The sound times are not controlled for the sound system. For the sounds to be repeated, the sound is continued until sound stop is used to request stopping. An application must be used to control the sound times. * For details about the repeat sounds, see 1.3 Relationship Between Sounds and Events. 		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_Play_Sound(unsigned int Ap_ID, _ELIB_SS_PLAY_SOUND *Play_Sound);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Play_Sound	_ELIB_SS_PLAY_SOUND *	I	Pointer to the sound start structure
Return value	Type	-	
ret	int	O	Play enabled: ELIB_SS_PLAY_OK Play disabled: ELIB_SS_PLAY_NG Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMEERR
Remark			

The sound start structure is as follows.

```
typedef struct tag_ELIB_SS_PLAY_SOUND { /* Sound start structure */
    unsigned int mode; /* [I/-] Sound type. See <mode setting type> below. */
    unsigned int Option; /* [I/-] Option information. See <Option information> below. */
    unsigned int Sound_Mode_Led; /* [I/-] LED color information when playing. See <LED setting type> below. */
    unsigned short SoundID; /* [I/-] The sound ID is stored. */
    char dummy[2]; /* [I/-] Boundary dummy iEunusediE */
} _ELIB_SS_PLAY_SOUND;
```

<mode setting type>

The <Sound type>, <Sound scene information>, and <Manner mode sound management> below must be specified for the second argument _ELIB_SS_PLAY_SOUND structure member mode. Specify by taking the logical add of <Sound type>, <Sound scene information>, and <Manner mode sound management> as shown below. See Use example for reference.

mode = <Sound type> | <Sound scene information> | <Manner mode sound management> ;

Sound type	Used scene
ELIB_SS_ALLTELMELO	All regular telephone ring tones
ELIB_SS_ORGMELO	Original ring tones only
ELIB_SS_ORGMELO3	Original ring tones only
ELIB_SS_SITEIMELO	Specified ring tone
ELIB_SS_MADRESS	Mail specified ring tone
ELIB_SS_MAIL	Ring tone when receiving mail
ELIB_SS_MESR	Ring tone when receiving a message request
ELIB_SS_MESF	Ring tone when receiving message free
ELIB_SS_ALARM	Schedule alarm sound
ELIB_SS_TODO	ToDo alarm sound
ELIB_SS_MELO_PLAYER	Melody player program play Plays the melodies registered using melody player list registration.
ELIB_SS_OTHER	Other effect sounds
ELIB_SS_184TELMELO	Non-notification setting ring tone
ELIB_SS_PUBTELMELO	Public telephone ring tone
ELIB_SS_IMPTELMELO	Notification disabled ring tone
ELIB_SS_VPMELO	Videophone ring tone
ELIB_SS_HORYU	Conversation on-hold tone
ELIB_SS_HORYU_DEMO	Conversation on-hold tone demo play
ELIB_SS_UD	Unrestricted digital ring tone
ELIB_SS_PPP	Packet (PPP) ring tone

<Sound scene information>

ELIB_SS_SOUND_SCENE_NORMAL /* Regular scene */
 ELIB_SS_SOUND_SCENE_EVENT /* Specify for the following scenes. */

List of scenes for which ELIB_SS_SOUND_SCENE_EVENT is specified
Starting ring tones when receiving a voice
Starting ring tones when receiving mail
Starting ring tones when specified ring tones have been set using a specified handy function when receiving a regular voice (specified ring tones)
Starting ring tones when mail specified ring tones have been set using a specified handy function when receiving mail (mail specified ring tones)
Starting ring tones when receiving a message request
Starting ring tones when receiving message free
Starting alarm sounds when playing schedule alarms
Starting ring tones when receiving non-notification setting
Starting alarm sounds when playing ToDo function alarms
Starting ring tones when receiving videophone
Unrestricted digital ring tone
Packet (PPP) ring tone

<Manner mode sound management>

ELIB_SS_SOUND_MANNER_NORMAL /* Based on the manner mode setting */
 ELIB_SS_SOUND_MANNER_FORCE /* When playing temporarily during manner mode */

Scenes for which ELIB_SS_SOUND_MANNER_FORCE is specified
Playing during manner mode by user confirmation using the melody player function
Playing during manner mode by user confirmation using the original melody function

<LED setting type>

The <LED color specification> and <Multifunction key backlight> below must be specified for the second argument _ELIB_SS_PLAY_SOUND structure member Sound_Mode_Led. Specify by taking the logical add of <LED color specification> and <Multifunction key backlight> as shown below. See Use example for reference.

Sound_Mode_Led = <LED color specification> | <Multifunction key backlight>

<LED color specification>

ELIB_SS_LED_COLOR_OFF /* The LED is not turned on. */
 ELIB_SS_LED_NO_SET /* The LED is turned on as described below based on the sound type specified for mode. */

```
ELIB_LMP_LED_COLOR1; /* Specify the illumination color defined using the ELIB LMP service. (*1)*/
/* The LED flashes using the illumination color set from the application. */
```

*1: This function is used when specified illumination or mail specified illumination has been set using the telephone directory or specified handy function. When playing using ELIB_SS_ALLTELMELO or ELIB_SS_SITEIMELO, the specified illumination information accessed by the data exchange service must be set for the LED color information (Sound_Mode_Led) if specified illumination has been set.

<LED operation when ELIB_SS_LED_NO_SET is specified>

Sound type (mode)	Flashing color
ELIB_SS_ALLTELMELO ELIB_SS_TELMELO ELIB_SS_MELO_PLAYER ELIB_SS_ORGMELO ELIB_SS_ORGMELO3 ELIB_SS_184TELMELO ELIB_SS_PUBTELMELO ELIB_SS_IMPTTELMELO ELIB_SS_SITEIMELO ELIB_SS_VPMELO ELIB_SS_UD	Conforms to the illumination color of voice receiving accessed by the LMP service.
ELIB_SS_MAIL ELIB_SS_MADRESS ELIB_SS_PPP	Conforms to the illumination color of i-mode mail accessed by the LMP service.
ELIB_SS_MESR	Conforms to the illumination color of message R accessed by the LMP service.
ELIB_SS_MESF	Conforms to the illumination color of message F accessed by the LMP service.
Other than the above	The illumination color flashes based on the specifications of the sound system.

<Multifunction key backlight>

```
ELIB_SS_SOUND_CROSSKEY_OFF /* Goes on and off synchronized with the key backlight. */
ELIB_SS_SOUND_CROSSKEY_ON /* Goes on and off synchronized with incoming LED flashing. */
* The setting of the multifunction key backlight must be decided using an application.
```

<Option information>

The option information must be set using the following bits (1 to 32).

Bit 1: Loop present/absent information (no loop: ELIB_SS_PLAY_LOOP_OFF, loop: ELIB_SS_PLAY_LOOP_ON;)

Bits 2 to 32: Reserved bits (Because these bits are reserved for future use, they must be initialized to 0.)

For a detailed description of the events, see 3.1 Event Formats.

Use example 1: When the sound scene is a regular ring tone selection function (menu 13).

```

    unsigned int Ap_ID; /* Application ID */
    _ELIB_SS_PLAY_SOUND Play_Sound; /* Sound ID of the ring tone to be played */
    int ret; /* Function return value */
    /** Parameter setting */
    Ap_ID = 1;
    Play_Sound.mode = ELIB_SS_ALLTELMELO | /*Sound type: Regular ring tone*/
                    ELIB_SS_SOUND_SCENE_NORMAL | /*Sound scene information: Regular scene*/
                    ELIB_SS_SOUND_MANNER_NORMAL; /*Manner mode sound management: Regular sound*/
    Play_Sound.SoundID = List_SoundID;
    Play_Sound.Option = ELIB_SS_PLAY_LOOP_ON; /* Loop */
    Play_Sound.Sound_Mode_Led= ELIB_SS_LED_NO_SET | /* The LED goes on based on the sound type
    setting. */
                    ELIB_SS_SOUND_CROSSKEY_ON; /* The multifunction key backlight goes on synchronized with
    the incoming LED. */

    ret = Elib_SS_Play_Sound(Ap_ID, &Play_Sound);

```

Use example 2: When the sound scene is regular incoming (no specified handy function setting).

```

    unsigned int Ap_ID; /* Application ID */
    _ELIB_SS_PLAY_SOUND Play_Sound; /* Sound ID of the ring tone to be played */
    int ret; /* Function return value */
    /** Parameter setting */
    Ap_ID = 1;
    Play_Sound.mode = ELIB_SS_ALLTELMELO | /*Sound type: Regular ring tone*/
                    ELIB_SS_SOUND_SCENE_EVENT | /*Sound scene information: Ring tone sound when
    receiving a regular voice*/
                    ELIB_SS_SOUND_MANNER_NORMAL; /*Manner mode sound management: Regular sound*/
    Play_Sound.SoundID = Set_SoundID;
    Play_Sound.Option = ELIB_SS_PLAY_LOOP_ON; /* Loop */
    Play_Sound.Sound_Mode_Led= ELIB_SS_LED_NO_SET | /* The LED goes on based on the sound type
    setting. */
                    ELIB_SS_SOUND_CROSSKEY_ON; /* The multifunction key backlight goes on synchronized with
    the incoming LED. */

    ret = Elib_SS_Play_Sound( Ap_ID, &Play_Sound);

```

Use example 3: When the sound scene is melody player and the manner is released temporarily during manner mode.

```

    unsigned int Ap_ID; /* Application ID */
    _ELIB_SS_PLAY_SOUND Play_Sound; /* Sound ID of the ring tone to be played */
    int ret; /* Function return value */
    /** Parameter setting */
    Ap_ID = 1;
    Play_Sound.mode = ELIB_SS_MELO_PLAYER | /*Sound type: Melody player*/

```

```

        ELIB_SS_SOUND_SCENE_NORMAL |    /*Sound scene:  Regular scene*/
        ELIB_SS_SOUND_SCENE_MANNER_FORCE; /* Manner mode sound management:  Playing
during manner mode by user confirmation using the melody player function*/
    Play_Sound.SoundID = 1;
    Play_Sound.Option = ELIB_SS_PLAY_LOOP_OFF; /* No loop */
    Play_Sound.Sound_Mode_Led= ELIB_SS_LED_NO_SET | /* The LED goes on based on the sound type
setting.*/
        ELIB_SS_SOUND_CROSSKEY_ON ;/* The multifunction key backlight goes on synchronized with
the incoming LED. */
    ret = Elib_SS_Play_Sound(Ap_ID, &Play_Sound);

```

Use example 4: When the sound scene is regular ring tone and a specified handy function is set (specified ring tone and specified illumination).

```

    unsigned int Ap_ID;    /* Application ID */
    _ELIB_SS_PLAY_SOUND Play_Sound; /* Sound ID of the ring tone to be played */
    int ret;               /* Function return value */
    /** Parameter setting */
    Ap_ID = 1;
    Play_Sound.mode = ELIB_SS_SITEIMELO /*Sound type; Specified ring tone*/
        ELIB_SS_SOUND_SCENE_EVENT | /* Sound scene information: Ring tone sound when a
specified handy function has been set using the specified handy function when receiving a regular ring tone
(specified ring tone)*/
        ELIB_SS_SOUND_MANNER_NORMAL; /*Manner mode sound management: Regular sound*/
    Play_Sound.SoundID = 1;
    Play_Sound.Option = ELIB_SS_PLAY_LOOP_ON; /* Loop */
    Play_Sound.Sound_Mode_Led= ELIB_LMP_LED_COLOR1 /* Illumination color accessed by a telephone
directory application */
        ELIB_SS_SOUND_CROSSKEY_ON /* The multifunction key backlight goes on synchronized with
the incoming LED. */
    ret = Elib_SS_Play_Sound( Ap_ID, &Play_Sound);

```

* Key confirmation sound restrictions of the melody player

During the sounds started using the sound types below, specific key confirmation sounds are not played because it is judged that the melody player is operating (specific key confirmation sounds: upper keys, lower keys, Clear key, and S2 key).

```

    ELIB_SS_ORGMELO          /* Original ring tones only*/
    ELIB_SS_MELO_PLAYER      /* Melody player program play
        The melodies registered using melody player list registration aplaybacked. */

```

9.44 Sound Stop

Classification	Functions for external processes of the sound system service		
Function	Sound stop	Symbol	Elib_SS_Stop_Sound
Functional overview	<p>Purpose: Requests stopping of the sound (ring tone, original melody, key sound, and so on).</p> <ul style="list-style-type: none"> - A sound ID specified for starting the ring tone must be specified for the sound ID. - The events for which notification has been requested using the sound system event occurrence notification request (Elib_SS_Request) are posted to the high-level processes. The following events are posted when the sound stops. For key sounds, however, no events are posted even if notification is requested. <p style="margin-left: 40px;">Melody sound interrupt notification (SoundNotify_STOPSOUND) Alarm sound interrupt notification (SoundNotify_ALARM_STOP)</p> <p style="margin-left: 40px;">* For details about the events, see 1.3 Relationship Between Sounds and Events.</p> <ul style="list-style-type: none"> - To release the notification request, use sound system event occurrence notification release (Elib_SS_Cancel). 		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_Stop_Sound(unsigned int Ap_ID, _ELIB_SS_STOP_SOUND *Stop_Sound);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Stop_Sound	_ELIB_SS_STOP_SOUND *	I	Pointer to the sound stop structure
Return value	Type	-	
Ret	int	O	Normal end: ELIB_SS_OK Abnormal end: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark			

The second argument sound stop structure is as follows.

```
typedef struct tag_ELIB_SS_STOP_SOUND {
    unsigned int    mode; /* [I/-] See <mode setting type> below. */
    unsigned short SoundID; /* [I/-] The sound ID is stored. When ELIB_SS_ALL (stop all
sounds) is specified for the mode, the SoundID is ignored. */
    char            dummy[2]; /* [-/-] Boundary dummy (unused) */
} _ELIB_SS_STOP_SOUND;
```

<mode setting type>

The <Sound type> and <Sound scene information> below must be specified for the structure member mode. Specify by taking the logical add of <Sound type> and <Sound scene information> as shown below. See Use example for reference.

mode = <Sound type> | <Sound scene> ;

A value specified for <Sound type> of Sound Start (F-3) must be specified.

Sound type	Used scene
ELIB_SS_ALL	Stop all sounds. (*1)
ELIB_SS_ALL1	Stop all sounds other than key sounds. (*1)
ELIB_SS_ALLTELMELO	All regular telephone ring tones
ELIB_SS_ORGMELO	Original ring tones only
ELIB_SS_ORGMELO3	Original ring tones only
ELIB_SS_SITEIMELO	Specified ring tone
ELIB_SS_MADRESS	Mail specified ring tone
ELIB_SS_MAIL	Ring tone when receiving mail
ELIB_SS_MESR	Ring tone when receiving a message request
ELIB_SS_MESF	Ring tone when receiving message free
ELIB_SS_ALARM	Schedule alarm sound
ELIB_SS_MELO_PLAYER	Melody player play
ELIB_SS_OTHER	Other effect sounds
ELIB_SS_184TELMELO	Non-notification setting ring tone
ELIB_SS_PUBTELMELO	Public telephone ring tone
ELIB_SS_IMPTELMELO	Notification disabled ring tone
ELIB_SS_VPMELO	Videophone ring tone
ELIB_SS_TODO	ToDo alarm sound
ELIB_SS_HORYU	Conversation on-hold tone
ELIB_SS_HORYU_DEMO	Conversation on-hold ring tone demo play
ELIB_SS_UD	Unrestricted digital ring tone
ELIB_SS_PPP	Packet (PPP) ring tone

<Sound scene information>

ELIB_SS_SOUND_SCENE_NORMAL /*Regular scene*/

ELIB_SS_SOUND_SCENE_EVENT /* Specify for the following scenes.*/

List of scenes for which ELIB_SS_SOUND_SCENE_EVENT is specified
--

	Stopping ring tones when receiving a regular voice	
	Stopping ring tones when receiving mail	
	Stopping ring tones when specified ring tones have been set using a specified handy function when receiving a regular voice (specified ring tones)	
	Stopping ring tones when mail specified ring tones have been set using a specified handy function when receiving mail (mail specified ring tones)	
	Stopping ring tones when receiving a message request	
	Stopping ring tones when receiving message free	
	Stopping alarm sounds when playing schedule alarms	
	Stopping ring tones when receiving non-notification setting	
	Stopping alarm sounds when playing ToDo function alarms	
	Stopping ring tones when receiving videophone	
	Unrestricted digital ring tone stop	
	Packet (PPP) ring tone stop	
- For sound IDs other than the above, see the IDs defined for use as other sounds in Appendix 3.3 Sound IDs .		

Note 1: When <Sound type> ELIB_SS_ALL and ELIB_SS_ALL1 are specified, the sound system does not manage conflicts when stopping the sounds. (The sounds are stopped unconditionally.) The calling side must handle any conflicts that may occur.

Use example 1: Stopping sounds when the sound stop scene is regular ring tone selection

```
unsigned int Ap_ID; /* Application ID */
_ELIB_SS_STOP_SOUND Stop_Sound; /* Sound stop structure */
int ret; /* Function return value */

/** Parameter setting */
Ap_ID = 1;
/* Specify the sound type and sound stop scene information. */
Stop_Sound.mode = ELIB_SS_ALLTELMELO | ELIB_SS_SOUND_SCENE_NORMAL ;
Stop_Sound.SoundID = 1;
ret = Elib_SS_Stop_Sound(Ap_ID, &Stop_Sound);
```

Use example 2: Stopping sounds when the sound stop scene is regular ring tone

```
unsigned int Ap_ID; /* Application ID */
_ELIB_SS_STOP_SOUND Stop_Sound; /* Sound stop structure */
int ret; /* Function return value */

/** Parameter setting */
Ap_ID = 1;
/* Specify the sound type and sound stop scene information. */
Stop_Sound.mode = ELIB_SS_ALLTELMELO | ELIB_SS_SOUND_SCENE_EVENT ;
Stop_Sound.SoundID = 1;
ret = Elib_SS_Stop_Sound(Ap_ID, &Stop_Sound);
```

Use example 3: Stopping all sounds other than key sounds

```
unsigned int Ap_ID; /* Application ID */
_ELIB_SS_STOP_SOUND Stop_Sound; /* Sound stop structure */
int ret; /* Function return value */

/** Parameter setting */
Ap_ID = 1;
/* Specify the sound type and sound stop scene information. */
Stop_Sound.mode = ELIB_SS_ALL1 | ELIB_SS_SOUND_SCENE_NORMAL ;
Stop_Sound.SoundID = 1;
ret = Elib_SS_Stop_Sound(Ap_ID, &Stop_Sound);
```

9.45 Videophone File Playback Start

Classification	Functions for external processes of the sound system service		
Function	Videophone file playback start	Symbol	Elib_SS_VideoConfFilePlay
Functional overview	<p>Purpose: Starts playing the videophone response hold, response message, or conversation on-hold tone.</p> <p>The following event is posted when videophone file playback is started.</p> <ul style="list-style-type: none"> - Videophone file playback start completed notification: SoundNotify_TVPLAYSTART_COMP (See 1.3 Relationship Between Sounds and Events.) <p>The following event is posted when videophone file playback is stopped naturally.</p> <ul style="list-style-type: none"> - Videophone file playback stop completed notification: SoundNotify_TVPLAYSTOP_COMP (See 1.3 Relationship Between Sounds and Events.) 		
Include file	srv_ss.h		
Calling sequence	Int Elib_SS_VideoConfFilePlay(unsigned int Ap_ID, unsigned char File_No);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
File_No	unsigned char	I	File playback type
Return value	Type	-	
ret	int	O	Normal: ELIB_SS_OK Abnormal: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>Specify one of the following for the second argument File_No.</p> <p>Videophone response on-hold tone: ELIB_SS_TV_RP_FIX_PAUSE_LOOP (fixed message)</p> <p>Videophone response on-hold tone: ELIB_SS_TV_RP_FIX_PAUSE_LOOP2 (fixed message)</p> <p>Videophone response on-hold tone: ELIB_SS_TV_REC_SPALSSTA_LOOP1 (unused)</p> <p>Videophone response on-hold tone: ELIB_SS_TV_REC_SPALSSTA_LOOP2 (unused)</p> <p>Videophone response message: ELIB_SS_TV_MSG_PLAY1 (fixed message 1)</p> <p>Videophone response message: ELIB_SS_TV_MSG_PLAY2 (fixed message 2)</p> <p>Videophone response message: ELIB_SS_TV_MSG_PLAY3 (fixed message 3)</p> <p>Videophone response message: ELIB_SS_TV_REC_SPALSTA1 (unused)</p> <p>Videophone response message: ELIB_SS_TV_REC_SPALSTA2 (unused)</p> <p>Videophone conversation on-hold tone: ELIB_SS_TV_HORU1 (default sound)</p> <p>Videophone conversation on-hold tone: ELIB_SS_TV_HORYUVOICE1 (unused)</p> <p>Videophone conversation on-hold tone: ELIB_SS_TV_HORYUVOICE2 (unused)</p> <p>Use example 1: Playing the videophone response on-hold tone (fixed message)</p>		


```
        unsigned int Ap_ID;           /* Application ID      */
int  ret;           /* Function return value */
unsigned char File_No;           /* File playback type */

/** Parameter setting */
Ap_ID = 1;
File_No = ELIB_SS_TV_RP_FIX_PAUSE_LOOP;
ret = Elib_SS_VideoConfFilePlay (Ap_ID, File_No);
```

9.46 Videophone File Playback Stop

Classification	Functions for external processes of the sound system service		
Function	Videophone file playback stop	Symbol	Elib_SS_VideoConfFilePlayStop
Functional overview	<p>Purpose: Stops playbacking the videophone response hold, response message, or conversation on-hold tone.</p> <p>The following event is posted when videophone file playback is stopped.</p> <ul style="list-style-type: none"> - Videophone file playback stop completed notification: SoundNotify_TVPLAYSTOP_COMP (See 1.3 Relationship Between Sounds and Events.) 		
Include file	srv_ss.h		
Calling sequence	int Elib_SS_VideoConfFilePlayStop(unsigned int Ap_ID);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Return value	Type	-	
ret	int	O	Normal: ELIB_SS_OK Abnormal: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>Note: This function stops the sound being played unconditionally when videophone file playback is started (Elib_SS_VideoConfFilePlay). The calling side must handle any conflicts that may occur.</p> <p>Use example 1: Stopping playback of videophone response hold (fixed message)</p> <pre> unsigned int Ap_ID; /* Application ID */ int ret; /* Function return value */ /** Parameter setting */ Ap_ID = 1; ret = Elib_SS_VideoConfFilePlayStop(Ap_ID); </pre>		

9.47 Videophone DTMF Start

Classification	Functions for external processes of the sound system service		
Function	Videophone DTMF start	Symbol	Elib_SS_VideoConf_DtmfStr
Functional overview	<p>Purpose: Plays the videophone DTMF (file playback). The event is posted after DTMF play is completed. Notification event: SoundNotify_TVDTMF_COMP (The event is not posted when DTMF is started.)</p>		
Include file	srv_ss.h		
Calling sequence	Int Elib_SS_VideoConf_DtmfStr(unsigned int Ap_ID, unsigned char Dtmf_Num);		
Argument	Type	I/O	Description
Ap_ID	unsigned int	I	Application ID
Dtmf_Num	unsigned char	I	DTMF number
Return value	Type	-	
ret	int	O	Normal: ELIB_SS_OK Abnormal: ELIB_SS_NG Parameter error: ELIB_SS_PRMERR
Remark	<p>One of the following must be specified for the second argument Dtmf_Num.</p> <p>DTMF sound (1): ELIB_SS_TV_DTMF1 DTMF sound (2): ELIB_SS_TV_DTMF2 DTMF sound (3): ELIB_SS_TV_DTMF3 DTMF sound (4): ELIB_SS_TV_DTMF4 DTMF sound (5): ELIB_SS_TV_DTMF5 DTMF sound (6): ELIB_SS_TV_DTMF6 DTMF sound (7): ELIB_SS_TV_DTMF7 DTMF sound (8): ELIB_SS_TV_DTMF8 DTMF sound (9): ELIB_SS_TV_DTMF9 DTMF sound (0): ELIB_SS_TV_DTMF0 DTMF sound (*): ELIB_SS_TV_DTMFAST DTMF sound (#): ELIB_SS_TV_DTMFSRP</p> <p>Use example 1:</p> <pre> unsigned int Ap_ID; /* Application */ int ret; /* Function return value */ unsigned char Dtmf_Num; /* DTMF number*/ /** Parameter setting **/ Ap_ID = 1; Dtmf_Num = ELIB_SS_TV_DTMF1; ret = Elib_SS_VideoConf_DtmfStr(Ap_ID, Dtmf_Num); </pre>		

10. SD File

10.1 SD management file service event registration

Classification	SD management file service support function		
Function	Event handler setting	Symbol	Elib_SDFS_Request
Function overview	<p>This function registers an application event for the SD management file service.</p> <p>- Target event examples</p> <p>SDFSNotify_CARD_REMOVE: Card removalSDFSNotify_CARD_CHECKQUALIFY_END: Termination of card qualify check</p> <p>SDFSNotify_FORMAT_END: Termination of format</p> <p>SDFSNotify_SAVE_END: Termination of data registration</p> <p>SDFSNotify_SAVE_DIR_END: Termination of folder registrationSDFSNotify_LOAD_END: Termination of data readSDFSNotify_CHKDSK_END: Termination of check disk</p> <p>SDFSNotify_DELETE_END: Termination of data deletion</p> <p>SDFSNotify_ALL_DELETE_END: Termination of deletion of all data items in folder</p> <p>SDFSNotify_DELETE_DIR_END: Termination of folder deletion</p> <p>SDFSNotify_REFRESH_DATAID: Update of management table</p> <p>SDFSNotify_COPY_END: Termination of data copying</p> <p>SDFSNotify_INFOGET_END: Termination of information acquisition</p> <p>SDFSNotify_LIST_INFOGET_END: Termination of list information acquisition</p> <p>SDFSNotify_vObject_LIST_INFOGET_END: Termination of vObject list information acquisition</p> <p>SDFSNotify_vObject_INFOGET_END: Termination of vObject information acquisition</p> <p>SDFSNotify_DPOF_Count_GET_END: Termination of acquisition of the number of pages with DPOF printing specified</p> <p>SDFSNotify_PIMIMPORT_END: Termination of SD-PIM import</p> <p>SDFSNotify_ALL_PIMIMPORT_END: Termination of SD-PIM total import</p> <p>SDFSNotify_PIMBACKUP_END: Termination of SD-PIM backup</p> <p>SDFSNotify_ALL_PIMBACKUP_END: Termination of SD-PIM total backup</p> <p>SDFSNotify_TITLECHG_END: Termination of title change</p> <p>SDFSNotify_CHG_SDSTATE: Report of SD icon status</p> <p>SDFSNotify_MOVE_END: Termination of data movement</p> <p>SDFSNotify_DPOF_Count_SET_END: Termination of setting of the number of pages with DPOF printing specified</p> <p>SDFSNotify_INFOSET_END: Termination of information setting</p> <p>SDFSNotify_INIT_END: Termination of folder initialization</p> <p>SDFSNotify_NOEXIST: Absence of SD card</p> <p>SDFSNotify_EXIST: Presence of SD card</p> <p>SDFSNotify_EXIST_WP: Presence of SD card (write protected)</p> <p>SDFSNotify_EXIST_ERR: Presence of SD card (error)</p> <p>SDFSNotify_ALL: All</p>		
Include file	srv_sdfs.h		
Calling sequence	Int32_t Elib_SDFS_Request(UINT Ap_ID, uint32_t EventCode ,MsbFunc Func);		

Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
EventCode	uint32_t	I	<p>SDFSNotify_CARD_REMOVE: Card removal</p> <p>SDFSNotify_CARD_CHECKQUALIFY_END: Termination of card qualify check</p> <p>SDFSNotify_FORMAT_END: Termination of format</p> <p>SDFSNotify_SAVE_END: Termination of data registration</p> <p>SDFSNotify_SAVE_DIR_END: Termination of folder registration</p> <p>SDFSNotify_LOAD_END: Termination of data read</p> <p>SDFSNotify_CHKDSK_END: Termination of check disk</p> <p>SDFSNotify_DELETE_END: Termination of data deletion</p> <p>SDFSNotify_ALL_DELETE_END: Termination of deletion of all data items in folder</p> <p>SDFSNotify_DELETE_DIR_END: Termination of folder deletion</p> <p>SDFSNotify_REFRESH_DATAID: Update of management table</p> <p>SDFSNotify_COPY_END: Termination of data copying</p> <p>SDFSNotify_INFOGET_END: Termination of information acquisition</p> <p>SDFSNotify_LIST_INFOGET_END: Termination of list information acquisition</p> <p>SDFSNotify_vObject_LIST_INFOGET_END: Termination of vObject list information acquisition</p> <p>SDFSNotify_vObject_INFOGET_END: Termination of vObject information acquisition</p> <p>SDFSNotify_DPOF_Count_GET_END: Termination of acquisition of the number of pages with DPOF printing specified</p> <p>SDFSNotify_PIMIMPORT_END: Termination of SD-PIM import</p> <p>SDFSNotify_ALL_PIMIMPORT_END: Termination of SD-PIM total import</p> <p>SDFSNotify_PIMBACKUP_END: Termination of SD-PIM backup</p> <p>SDFSNotify_ALL_PIMBACKUP_END: Termination of SD-PIM total backup</p> <p>SDFSNotify_TITLECHG_END: Termination of title</p>

			change SDFSNotify_CHG_SDSTATE: Report of SD icon status SDFSNotify_MOVE_END: Termination of data movement SDFSNotify_DPOF_Count_SET_END: Termination of setting of the number of pages with DPOF printing specified SDFSNotify_INFOSET_END: Termination of information setting SDFSNotify_INIT_END: Termination of folder initialization SDFSNotify_NOEXIST: Absence of SD card SDFSNotify_EXIST: Presence of SD card SDFSNotify_EXIST_WP: Presence of SD card (write protected) SDFSNotify_EXIST_ERR: Presence of SD card (error) SDFSNotify_ALL: All
Func	MsbFunc	I	Pointer to function called back for event occurrence
Return value	Type	-	Description
Ret	Int32_t	O	ELIB_SDFS_RETOK: Normal end ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_MISC: Other errors
Remarks			
<div>- Types of functions that called back for event occurrence void func(MsbObject *client, MsbObject *server, void *sendData, void *recvData, void *data, MsbEnvironment *ev);</div> <div>- Data sent for absence of SD card or presence of SD card (normal/write protected/error) is listed below (info, subinfo = information in event transmission data).</div>			
Event symbol	info (transmission data)	subinfo (transmission data)	Explanation
SDFSNotify_NOEXIST	SDFSNotify_NOEXIST	SDFS_RETOK	Absence of SD card
SDFSNotify_EXIST	SDFSNotify_EXIST	SDFS_RETOK	Presence of SD card
SDFSNotify_EXIST_WP	SDFSNotify_EXIST_WP	SDFS_RETOK	Presence of SD card (write protected)
SDFSNotify_EXIST_ERR	SDFSNotify_EXIST_ERR	SDFS_RETOK	Presence of SD card (error)



10.2 SD management file service event release

Classification	SD management file service support function		
Function	Event occurrence report release	Symbol	Elib_SDFS_Cancel
Function overview	<p>This function releases registered application events and reported events.</p> <p>- Target event examples</p> <p>Same as SD management file service event registration (10.1)</p>		
Include file	srv_sdfs.h		
Calling sequence	Int32_t Elib_SDFS_Cancel(UINT Ap_ID, uint32_t EventCode);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
EventCode	uint32_t	I	<p>SDFSNotify_CARD_REMOVE: Card removal</p> <p>SDFSNotify_CARD_CHECKQUALIFY_END: Termination of card qualify check</p> <p>SDFSNotify_FORMAT_END: Termination of format</p> <p>SDFSNotify_SAVE_END: Termination of data registration</p> <p>SDFSNotify_SAVE_DIR_END: Termination of folder registration</p> <p>SDFSNotify_LOAD_END: Termination of data read</p> <p>SDFSNotify_CHKDSK_END: Termination of check disk</p> <p>SDFSNotify_DELETE_END: Termination of data deletion</p> <p>SDFSNotify_ALL_DELETE_END: Termination of deletion of all data items in folder</p> <p>SDFSNotify_DELETE_DIR_END: Termination of folder deletion</p> <p>SDFSNotify_REFRESH_DATAID: Update of management table</p> <p>SDFSNotify_COPY_END: Termination of data copying</p> <p>SDFSNotify_INFOGET_END: Termination of information acquisition</p> <p>SDFSNotify_LIST_INFOGET_END: Termination of list information acquisition</p> <p>SDFSNotify_vObject_LIST_INFOGET_END: Termination of vObject list information acquisition</p>

			<p>SDFSNotify_vObject_INFOGET_END: Termination of vObject information acquisition</p> <p>SDFSNotify_DPOF_Count_GET_END: Termination of acquisition of the number of pages with DPOF printing specified</p> <p>SDFSNotify_PIMIMPORT_END: Termination of SD-PIM import</p> <p>SDFSNotify_ALL_PIMIMPORT_END: Termination of SD-PIM total import</p> <p>SDFSNotify_PIMBACKUP_END: Termination of SD-PIM backup</p> <p>SDFSNotify_ALL_PIMBACKUP_END: Termination of SD-PIM total backup</p> <p>SDFSNotify_TITLECHG_END: Termination of title change</p> <p>SDFSNotify_CHG_SDSTATE: Report of SD icon status</p> <p>SDFSNotify_MOVE_END: Termination of data movement</p> <p>SDFSNotify_DPOF_Count_SET_END: Termination of setting of the number of pages with DPOF printing specified</p> <p>SDFSNotify_INFOSET_END: Termination of information setting</p> <p>SDFSNotify_INIT_END: Termination of folder initialization</p> <p>SDFSNotify_NOEXIST: Absence of SD card</p> <p>SDFSNotify_EXIST: Presence of SD card</p> <p>SDFSNotify_EXIST_WP: Presence of SD card (write protected)</p> <p>SDFSNotify_EXIST_ERR: Presence of SD card (error)</p> <p>SDFSNotify_ALL: All</p>
Return value	Type	-	Description
Ret	int32_t	O	<p>ELIB_SDFS_RETOK: Normal end</p> <p>ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error</p> <p>ELIB_SDFS_RETERR_MISC: Other errors</p>
<p>Remark</p> <p>* <u>Each application should use this API before termination to release all event of which report are requested.</u></p>			

10.3 Processing halt (asynchronous)

Classification	SD management file service support function		
Function	SD card processing stop	Symbol	Elib_SDFS_AbortAsync
Function overview	<p>This function halts SD card processing of the SD management file service.</p> <ul style="list-style-type: none"> - Processing of an asynchronous function is halted. - This function operates asynchronously (terminates by reporting processing request halt to a function under process). - After this function is called, the target asynchronous function under process is halted (stopped) and then, the halt of processing is reported with an event message (event transmission data). This event message is issued by setting SDFSNotify_Abort as "info" value in transmission data and ELIB_SDFS_ABORTCALLED (negative value) as "subinfo" value. - No event is caused by this function. - If halt of write processing is requested, files with data has been written are deleted (scratch file deletion). 		
Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_AbortAsync(UINT Ap_ID);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
Return value	Type	-	Description
Ret	int32_t	O	<ul style="list-style-type: none"> - For normal end ELIB_SDFS_RETOK: Normal end (halt request accepted) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_MISC: Other errors
Remark			

10.4 File read

Classification	SD management file service support function		
Function	File read	Symbol	Elib_SDFS_LoadFromFile
Function overview	<p>This function reads a specified file and stores it in a buffer.</p> <ul style="list-style-type: none"> - A file specified with FileType and DataID is read and stored in a buffer specified with a buffer pointer (the buffer area should be allocated by the caller). - Specify a value obtained with Elib_SDFS_GetInfoList in DataID. - An error is returned if the file with a specified DataID is not registered (not present). - This function does not support PIM data. - This function operates synchronously (in other words, it does not return unless its processing terminates). 		
Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_LoadFromFile(UINT Ap_ID, uint16_t FileType, uint16_t DataID, uint8_t*Buffer, uint32_t SizeOfBuffer, uint32_t Offset);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
FileType	uint16_t	I	This argument specifies a file type. ELIB_SDFS_VIDEO : SD save dynamic image ELIB_SDFS_PICTURE : SD save static image PIM data is not supported.
DataID	uint16_t	I	This argument specifies a data ID. (10to16393)
Buffer	uint8_t*	I	This argument specifies the pointer to a storage buffer. The caller need allocate the buffer.
SizeOfBuffer	uint32_t	I	This argument specifies a buffer size.
Offset	uint32_t	I	This argument specifies a read starting offset (unit: byte). Specify 0 when data read is to start with the top of a file.
Return value	Type	-	Description
Ret	int32_t	O	<ul style="list-style-type: none"> - For normal end Read size (0<=ret<=Size:buffer-size) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NODIR: File not found ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error

		ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_RETERR_ABORTCALLED: Halt with halt request
Remark		

10.5 File registration

Classification	SD management file service support function		
Function	File registration	Symbol	Elib_SDFS_SaveToFile
Function overview	<p>This function saves data in a file.</p> <ul style="list-style-type: none"> - Data stored in a buffer is saved in a file. - When DataID is specified, data is overwritten on a file with the specified ID. - Specify a value obtained with the Elib_SDFS_GetInfoList function in Data ID. - When DataID=0 is specified, a file is created to save data. - An error is returned if data can not be saved due to an area shortage. - This function does not support PIM data. - This function operates synchronously (in other words, it does not return unless its processing terminates). - Note that there are rules on combination of the DataID, Option, and Offset arguments. 		
Include file	srv_sdfs.h		
Calling sequence	<pre>int32_t Elib_SDFS_SaveToFile(UINT Ap_ID, uint16_t FileType, uint16_t DataID, const uint8_t *Buffer, uint32_t SizeOfBuffer, uint8_t *NewTitleName, uint32_t Offset, int32_t Option);</pre>		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
FileType	uint16_t	I	ELIB_SDFS_PICTURE_JPEG: Save of static image with JPEG format ELIB_SDFS_VIDEO_ASF: Save of dynamic image with ASF format ELIB_SDFS_VIDEO_3GP: Save of dynamic image with 3GP format ELIB_SDFS_VIDEO_SDV: Save of dynamic image with SDV format ELIB_SDFS_VIDEO_MP4: Save of dynamic image with MP4 format PIM data is not supported.
DataID	uint16_t	I	This argument specifies a data ID. Call the function by specifying DataID=0 to save new data. To save data by updating saved data, specify a value not less than 1.
Buffer	const uint8_t*	I	This argument specifies a pointer to the top of a data buffer.
SizeOfBuffer	uint32_t	I	This argument specifies the size of data to be saved.
NewTitleName	uint8_t*	I	This argument specifies a title name for a file to be

			created. The caller should allocate the title field. Up to 64 bytes ending with null can be used.		
Offset	uint32_t	l	This argument specifies a write starting offset (unit: byte). Specify 0 when data write is to start with the top of a file.		
Option	int32_t	l	This argument specifies a write attribute. Only new data overwriting is supported. ELIB_SDFS_SAVEFILE_NEW: File creation or (new) overwriting If a file is present, overwriting after deleting data in the file.		
Return value	Type	-	Description		
Ret	int32_t	0	- For normal end Specified data ID - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NODIR: Folder not found ELIB_SDFS_RETERR_WRITEPROTECT: Write protect error ELIB_SDFS_RETERR_FILENUMMAX: The maximum number of files that can be registered is exceeded. ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_RETERR_ABORTCALLED: Halt with halt request ELIB_SDFS_RETERR_MEMFULL: Memory filled		
Remarks			<div>* Note that some combinations of the DataID and Option arguments are inhibited. This Platform supports only ELIB_SDFS_SAVEFILE_NEW and DataID:0.</div> <div>* For the Option and Offset arguments, only ELIB_SDFS_SAVEFILE_NEW and 0 are supported.</div> <div><div>- Rules on combination of the Option and DataID arguments.</div><div>0 (file creation)</div></div>		
<table><tr><td>Option/DataID</td><td>0 (file creation)</td><td>Other than 0 (allocated file)</td></tr></table>				Option/DataID	0 (file creation)
Option/DataID	0 (file creation)	Other than 0 (allocated file)			

ELIB_SDFS_SAVEFILE_NEW	New data save	Data overwriting (after deleting existing data)
ELIB_SDFS_SAVEFILE_APPEND	Inhibited	Data addition
ELIB_SDFS_SAVEFILEOVER_WRITE	Inhibited	Partial rewriting

- Combinations of the Data ID, Option, and Offset arguments for basic conditions

Condition	DataID	Option	Offset
File creation	0	ELIB_SDFS_SAVEFILE_NEW { ELIB_SDFS_DCFC} { ELIB_SDFS_DCFC_CPON} * ELIB_SDFS_DCFCxxxx can be specified only when ELIB_SDFS_PICTURE_JPEG is specified in FileType.	0
Data write at the end of an allocated file (data addition)	Other than 0	ELIB_SDFS_SAVEFILE_APPEND	0
Partial rewriting in an existing file	Other than 0	ELIB_SDFS_SAVEFILE_OVERWRITE	Rewriting start position

If a combination other than ELIB_SDFS_SAVEFILE_NEW and Offset=0 or ELIB_SDFS_SAVEFILE_APPEND and offset=0 is specified, the contents of data (padding) saved between the top or end of the file and the offset are not unique (but random).

10.6 File deletion

Classification	SD management file service support function		
Function	File deletion	Symbol	Elib_SDFS_DeleteFile
Function overview	<p>This function deletes a file specified with DataID.</p> <ul style="list-style-type: none"> - Specify a value obtained with the Elib_SDFS_GetInfoList function in Data ID. - The caller should allocate the array storing a specified ID. - An error is returned if a file with a specified ID is not registered. - This function operates synchronously (in other words, it does not return unless its processing terminates). 		
Include file	srv_sdfs.h		
Calling sequence	<pre>int32_t Elib_SDFS_DeleteFile(UINT Ap_ID, uint16_t FileType, uint16_t *DataID, uint16_t DeleteNumber, uint16_t *DeleteEndNumber);</pre>		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
FileType	uint16_t	I	ELIB_SDFS_VIDEO: Dynamic image ELIB_SDFS_PICTURE: Static image ELIB_SDFS_PIM:PIM
DataID	uint16_t*	I	This argument specifies the pointer to an array where the data ID of a file to be deleted is stored.
DeleteNumber	uint16_t	I	This argument specifies the number of files to be deleted.
DeleteEndNumber	uint16_t*	O	This argument specifies the number of deleted files.
Return value	Type	-	Description
Ret	int32_t	O	<ul style="list-style-type: none"> - For normal end ELIB_SDFS_RETOK - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NOFILE: File not found ELIB_SDFS_RETERR_WRITEPROTECT: Write protect error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_RETERR_ABORTCALLED: Halt with halt request
Remark			

- The caller can register a new file without using the SD file service by using the FullPathName variable in the list to directly call the extended file API.
- Information set in the list must not be changed.
- After this function terminates normally, other SD file service functions returns ELIB_SDFS_RETERR_BLOCKING to the caller unless the Elib_SDFS_CloseNewFileInfo function is called.

Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_OpenNewFileInfo(UINT Ap_ID, uint16_t FileType, ELIB_SDFS_FILEINFO *List);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
FileType	uint16_t	I	ELIB_SDFS_PICTURE_JPEG: JPEG image ELIB_SDFS_VIDEO_ASF: ASF dynamic image ELIB_SDFS_VIDEO_3GP: 3GP dynamic image ELIB_SDFS_VIDEO_SDV: SDV dynamic image ELIB_SDFS_VIDEO_MP4: MP4 dynamic image
List	ELIB_SDFS_FILEINFO *	O	Pointer to file detailed information structure
Return value	Type	-	Description
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK: Normal end (halt request accepted) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_WRITEPROTECT: Write protect error ELIB_SDFS_RETERR_FILENUMMAX: The maximum number of files that can be registered is exceeded. ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_RETERR_MEMFULL: Memory filled ELIB_SDFS_RETERR_ABORTCALLED:

			Halt with halt request
Remark	See 10.8, "New file information close".		

10.8 New file information close

Classification	SD management file service support function		
Function	New file information close	Symbol	Elib_SDFS_CloseNewFileInfo
Function overview	<p>This function registers new file name provided for an application in a management table.</p> <ul style="list-style-type: none"> - This function updates information about a file with a name obtained with the Elib_SDFS_LoadFromFileAsync function (10.10) in a management table. - This function also releases a lock set with the open function. - An application need always call this function after terminating file creation (writing) by normally obtaining a file name with the Elib_SDFS_OpenNewFileInfo function. - Specify information obtained with the Elib_SDFS_OpenNewFileInfo function in the list. If a value other than this is set or the list is modified, SD management file service operation is not guaranteed. - Set the size of a file where data is written in a list member. - If registration is to be canceled due to unsuccessful writing, call this function (by specifying 0 in Accept) after terminating closing of the opened file or deletion of the file with data written. 		
Include file	srv_sdfs.h		
Calling sequence	<pre>int32_t Elib_SDFS_CloseNewFileInfo(UINT Ap_ID, uint16_t FileType, const ELIB_SDFS_FILEINFO *List, int32_t Accept);</pre>		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
FileType	uint16_t	I	This argument specifies a file type.
List	ELIB_SDFS_FILEINFO *	I	This argument specifies the pointer to file detailed information structure.
Accept	int32_t	I	Specify a value other than 0 when writing in a file terminates normally (as a result, the file is registered in the management table). Specify 0 when writing in a file terminates abnormally (as a result, the file is not registered in the management table).
Return value	Type	-	Description
Ret	int32_t	O	<ul style="list-style-type: none"> - For normal end ELIB_SDFS_RETOK: Normal end - For abnormal end (negative numeric) ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_MISC: Other

			errors ELIB_SDFS_RETERR_MEMFULL: Memory filled
Remark			

10.9 Save destination folder set

Classification	SD management file service support function		
Function	Save destination folder set	Symbol	Elib_SDFS_SetSaveDir
Function overview	<p>This function sets a save destination folder.</p> <ul style="list-style-type: none"> - This function specifies a folder to be saved in an SD card. Setting with this function is valid for both the static image and dynamic image. - Specify a value obtained with the Elib_SDFS_GetInfoList function in DataID. - An error is returned if a file with a specified ID is not registered (is not present). - A folder with files as many as the maximum number (9,999 files) cannot be set. - This function does not support PIM data. A parameter error occurs if PIM is specified in DataID. - This function operates synchronously (in other words, it does not return unless its processing terminates). 		
Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_SetSaveDir(UINT Ap_ID, uint16_t DirType, uint16_t DataID);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
DirType	uint16_t	I	This argument specifies a file type. ELIB_SDFS_VIDEO : SD save dynamic image ELIB_SDFS_PICTURE : SD save static image PIM data is not supported.
DataID	uint16_t	I	This argument specifies a data ID.
Return value	Type	-	Description
Ret	int32_t	O	<ul style="list-style-type: none"> - For normal end ELIB_SDFS_RETOK - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NODIR: Folder not found ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_RETERR_ABORTCALLED: Halt with halt request ELIB_SDFS_RETERR_FILENUMMAX: The maximum number of files that can be registered is exceeded (folder cannot be set).
Remark			



10.10 File read (asynchronous)

Classification	SD management file service support function																								
Function	File read (asynchronous)	Symbol	Elib_SDFS_LoadFromFileAsync																						
Function overview	This function reads a file (asynchronously).																								
<div>- This function is an asynchronous version of the Elib_SDFS_LoadFromFile function (10.4).</div> <div>- Specify a value obtained with the Elib_SDFS_GetInfoList function in DataID.</div> <div>- This function operates asynchronously.</div> <div>- No event occurs unless this function terminates normally (processing starts).</div> <div>After terminating file read, this function creates and sends events listed below (info, subinfo = information in event transmission data).</div> <table><tr><th>Event symbol</th><th>info (transmission data)</th><th>subinfo (transmission data)</th><th>Explanation</th></tr><tr><td>SDFSNotify_LOAD_END</td><td>SDFSNotify_LOAD_END</td><td>SDFS_RETOK</td><td>Normalend</td></tr><tr><td>SDFSNotify_LOAD_END</td><td>SDFSNotify_LOAD_END</td><td>Error type (negative)</td><td>Abnormal end</td></tr></table> <div>- Error types are listed below.</div> <table><tr><th>Symbol</th><th>Type</th></tr><tr><td>ELIB_SDFS_RETERR_NOFILE</td><td>No file</td></tr><tr><td>ELIB_SDFS_RETERR_IO</td><td>I-O error</td></tr><tr><td>ELIB_SDFS_RETERR_ABORTCALLED</td><td>Halt with processing halt request</td></tr><tr><td>ELIB_SDFS_RETERR_MISC</td><td>Others</td></tr></table>				Event symbol	info (transmission data)	subinfo (transmission data)	Explanation	SDFSNotify_LOAD_END	SDFSNotify_LOAD_END	SDFS_RETOK	Normalend	SDFSNotify_LOAD_END	SDFSNotify_LOAD_END	Error type (negative)	Abnormal end	Symbol	Type	ELIB_SDFS_RETERR_NOFILE	No file	ELIB_SDFS_RETERR_IO	I-O error	ELIB_SDFS_RETERR_ABORTCALLED	Halt with processing halt request	ELIB_SDFS_RETERR_MISC	Others
Event symbol	info (transmission data)	subinfo (transmission data)	Explanation																						
SDFSNotify_LOAD_END	SDFSNotify_LOAD_END	SDFS_RETOK	Normalend																						
SDFSNotify_LOAD_END	SDFSNotify_LOAD_END	Error type (negative)	Abnormal end																						
Symbol	Type																								
ELIB_SDFS_RETERR_NOFILE	No file																								
ELIB_SDFS_RETERR_IO	I-O error																								
ELIB_SDFS_RETERR_ABORTCALLED	Halt with processing halt request																								
ELIB_SDFS_RETERR_MISC	Others																								
Include file	srv_sdfs.h																								
Calling sequence	int32_t Elib_SDFS_LoadFromFileAsync(UINT Ap_ID, uint16_t FileType, uint16_t DataID, uint32_t SizeOfBuffer, uint32_t Offset);																								
Argument	Type	I/O	Description																						
Ap_ID	UINT	I	Application ID																						
FileType	uint16_t	I	This argument specifies a file type. ELIB_SDFS_VIDEO: SD save dynamic image ELIB_SDFS_PICTURE: SD save static image																						
DataID	uint16_t	I	This argument specifies data number. (10to16393)																						
SizeOfBuffer	uint32_t	I	This argument specifies a buffer size.																						
Offset	uint32_t	I	This argument specifies a read starting offset (unit: byte). Specify 0 when data read is to start with the top of a file.																						
Return value	Type	-	Description																						

Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK: Normal end (processing start) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NODIR: File not found ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing)
Remark			
* See 10.4, “File read” (Elib_SDFS_LoadFromFile).			

10.11 File registration (asynchronous)

Classification	SD management file service support function		
Function	File registration	Symbol	Elib_SDFS_SaveToFileAsync
Function overview	This function reads data from a buffer and saves it in a file (asynchronously).		
<div><div></div><div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><</div></div></div>			

			<p>with ASF format</p> <p>ELIB_SDFS_VIDEO_3GP: Save of dynamic image with 3GP format</p> <p>ELIB_SDFS_VIDEO_SDV: Save of dynamic image with SDV format</p> <p>ELIB_SDFS_VIDEO_SDV: Save of dynamic image with MP4 format</p> <p>ELIB_SDFS_PICTURE_JPEG: Save of static image with JPEG format</p>
DataID	uint16_t	l	<p>This argument specifies a data ID.</p> <p>To save data by updating saved data, specify a value not less than 1. Call the function by specifying DataID=0 to save new data.</p>
Buffer_shmid	int	l	<p>This argument specifies the shared memory ID of a shard memory where a data buffer is stored.</p>
SizeOfBuffer	uint32_t	l	<p>This argument specifies the size of data to be saved.</p>
NewTitleName	uint8_t*	l	<p>This argument specifies a title name for a file to be created.</p> <p>The caller should allocate the title field.</p> <p>Up to 64 bytes ending with null can be used.</p>
Offset	uint32_t	l	<p>This argument specifies a write starting offset (unit: byte).</p> <p>Specify 0 when data write is to start with the top of a file.</p>
Option	int32_t	l	<p>This argument specifies a write attribute.</p> <p>Specify one of the following values:</p> <p>ELIB_SDFS_SAVEFILE_NEW:</p> <p>File creation or (new) overwriting (If a file is present, overwriting after deleting data in the file)</p> <p>ELIB_SDFS_SAVEFILE_APPEND:</p> <p>Data addition to an existing file.</p> <p>* This value cannot be specified with DataID=0 specified.</p> <p>ELIB_SDFS_SAVEFILE_OVERWRITE:</p> <p>Overwriting in an existing file (value to be specified for partial modification)</p> <p>* This value cannot be specified with DataID=0</p>

			<p>specified.</p> <p>When JPEG is specified in FileType, the following can also be specified (a value obtained in OR operation with one of the above values can be specified).</p> <p>ELIB_SDFS_SAVEFILE_DCFC: If the target static image data does not contain a thumbnail layout, create a thumbnail layout same as in JFIF(JPEG) data stored in a buffer indicated by the Buffer pointer and save the image data in a JPEG file conforming to the DCF standard by combining it with the created layout. Clear the Copyright flag for the created image.</p> <p>ELIB_SDFS_SAVEFILE_DCFC_CPON: If the target static image data does not contain a thumbnail layout, create a thumbnail layout same as in JFIF(JPEG) data stored in a buffer indicated by the Buffer pointer and save the image data in a JPEG file conforming to the DCF standard by combining it with the created layout. Set the Copyright flag for the created image.</p> <p>* Neither of the above values is specified, the contents of a buffer passed using an argument are output directly as a file.</p>
Return value	Type	-	Description
Ret	int32_t	O	<p>- For normal end ELIB_SDFS_RETOK: Normal end (processing start) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NODIR: File not found ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_WRITEPROTECT: Write protect error ELIB_SDFS_RETERR_ABORTCALLED: Halt with halt request</p>
Remark			

10.12 File deletion (asynchronous)

Classification	SD management file service support function		
Function	File deletion	Symbol	Elib_SDFS_DeleteFileAsync
Function overview	This function deletes a file specified with argument DataID (asynchronously).		
<div><div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div></div>			

DataID	uint16_t*	I	Data ID (deletion target file array)
DeleteNumber	uint16_t	I	Number of files to be deleted
Return value	Type	-	Description
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK: Normal end (processing start) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NOFILE: File not found ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_WRITEPROTECT: Write protect error
Remark			
* See 10.6, “File deletion” (Elib_SDFS_DeleteFile function).			

10.13 Delete all files in folder

Classification	SD management file service support function														
Function	Delete all files in folder (asynchronous)	Symbol	Elib_SDFS_DeleteAllFilesAsync												
Function overview	This function deletes all files in a folder specified with argument DataID.														
<ul style="list-style-type: none">- Specify a value obtained with the Elib_SDFS_GetInfoList function in DataID.- An error is returned if a folder with a specified ID is not registered.- The target folder need be initialized (Elib_SDFS_DirInitAsync).- This function operates asynchronously.- No event occurs unless this function terminates normally (processing starts) (return value: ELIB_SDFS_RETOK).															
After terminating file deletion, this function creates and sends events listed below (info, subinfo = additional information in event transmission data).															
<table><tr><th>Event symbol</th><th>info (transmission data)</th><th>subinfo (transmission data)</th><th>Explanation</th></tr><tr><td>SDFSNotify_ALL_DELETE_END</td><td>SDFSNotify_ALL_DELETE_END</td><td>ELIB_SDFS_RETOK</td><td>Normal end</td></tr><tr><td>SDFSNotify_ALL_DELETE_END</td><td>SDFSNotify_ALL_DELETE_END</td><td>Error type (negative)</td><td>Abnormal end</td></tr></table>				Event symbol	info (transmission data)	subinfo (transmission data)	Explanation	SDFSNotify_ALL_DELETE_END	SDFSNotify_ALL_DELETE_END	ELIB_SDFS_RETOK	Normal end	SDFSNotify_ALL_DELETE_END	SDFSNotify_ALL_DELETE_END	Error type (negative)	Abnormal end
Event symbol	info (transmission data)	subinfo (transmission data)	Explanation												
SDFSNotify_ALL_DELETE_END	SDFSNotify_ALL_DELETE_END	ELIB_SDFS_RETOK	Normal end												
SDFSNotify_ALL_DELETE_END	SDFSNotify_ALL_DELETE_END	Error type (negative)	Abnormal end												
<ul style="list-style-type: none">- Error types are listed below.															
<table><tr><th>Symbol</th><th>Type</th></tr><tr><td>ELIB_SDFS_RETERR_NODIR</td><td>No folder</td></tr><tr><td>ELIB_SDFS_RETERR_ILLEGAL_PARAM</td><td>Parameter error</td></tr></table>				Symbol	Type	ELIB_SDFS_RETERR_NODIR	No folder	ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error						
Symbol	Type														
ELIB_SDFS_RETERR_NODIR	No folder														
ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error														

ELIB_SDFS_RETERR_IO		I-O error	
ELIB_SDFS_RETERR_ABORTCALLED		Halt with halt request	
ELIB_SDFS_RETERR_MISC		Others	
Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_DeleteAllFilesAsync(UINT Ap_ID, uint16_t FileType, uint16_t DataID, uint8_t mode);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
FileType	uint16_t	I	ELIB_SDFS_VIDEO: SD save dynamic image ELIB_SDFS_PICTURE: SD save static image ELIB_SDFS_PIM: SD save PIM
DataID	uint16_t	I	This argument specifies a data ID (folder). For PIM data, specify a virtual root folder as follows: ELIB_SDFS_ROOTDIR_CARD:vCard ELIB_SDFS_ROOTDIR_CALEDAR:vCalendar ELIB_SDFS_ROOTDIR_INBOX: Received mail ELIB_SDFS_ROOTDIR_OUTBOX: Untransmitted mail ELIB_SDFS_ROOTDIR_SENTBOX: Transmitted mail ELIB_SDFS_ROOTDIR_NOTE: Free memorandum ELIB_SDFS_ROOTDIR_BOOKMARK: Bookmark
mode	uint8_t	I	This argument specifies deletion mode as follows: ELIB_SDFS_DELALL : Deletion of all files (that are listed) in a folder ELIB_SDFS_DELNOTRO : Deletion of files (that are listed) other than read-only file
Return value	Type	-	Description
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK: Normal end (processing start) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NODIR: Folder not found ELIB_SDFS_RETERR_NOTINIT: Flder not initialized ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_WRITEPROTECT: Write protect error

Remark	

10.14 Folder registration (asynchronous)

Classification	SD management file service support function																
Function	Folder registration	Symbol	Elib_SDFS_SaveToDirAsync														
Function overview	This function creates a folder under a folder specified with argument DataID (asynchronously).																
<div>- This function does not support PIM data..</div> <div>- This function operates asynchronously.</div> <div>- No event occurs unless this function terminates normally (processing starts) (return value: ELIB_SDFS_RETOK).</div>																	
After terminating folder registration, this function creates and sends events listed below (info, subinfo = information in event transmission data).																	
<table><tr><td>SDFSNotify_SAVE_DIR_END</td><td>info (transmission data)</td><td>subinfo (transmission data)</td><td>Explanation</td></tr><tr><td>SDFSNotify_SAVE_DIR_END</td><td>SDFSNotify_SAVE_DIR_END</td><td>Specified data ID</td><td>Normal end</td></tr><tr><td>SDFSNotify_SAVE_DIR_END</td><td>SDFSNotify_SAVE_DIR_END</td><td>Error type (negative)</td><td>Abnormal end</td></tr></table>				SDFSNotify_SAVE_DIR_END	info (transmission data)	subinfo (transmission data)	Explanation	SDFSNotify_SAVE_DIR_END	SDFSNotify_SAVE_DIR_END	Specified data ID	Normal end	SDFSNotify_SAVE_DIR_END	SDFSNotify_SAVE_DIR_END	Error type (negative)	Abnormal end		
SDFSNotify_SAVE_DIR_END	info (transmission data)	subinfo (transmission data)	Explanation														
SDFSNotify_SAVE_DIR_END	SDFSNotify_SAVE_DIR_END	Specified data ID	Normal end														
SDFSNotify_SAVE_DIR_END	SDFSNotify_SAVE_DIR_END	Error type (negative)	Abnormal end														
<div>- Error types are listed below.</div> <table><tr><td>Symbol</td><td>Type</td></tr><tr><td>ELIB_SDFS_RETERR_ILLEGAL_PARAM</td><td>Parameter error</td></tr><tr><td>ELIB_SDFS_RETERR_FILENUMMAX</td><td>Maximum directory or file number reached</td></tr><tr><td>ELIB_SDFS_RETERR_IO</td><td>I-O error</td></tr><tr><td>ELIB_SDFS_RETERR_ABORTCALLED</td><td>Halt with processing halt request</td></tr><tr><td>ELIB_SDFS_RETERR_MEMFULL</td><td>Memory filled</td></tr><tr><td>ELIB_SDFS_RETERR_MISC</td><td>Others</td></tr></table>				Symbol	Type	ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error	ELIB_SDFS_RETERR_FILENUMMAX	Maximum directory or file number reached	ELIB_SDFS_RETERR_IO	I-O error	ELIB_SDFS_RETERR_ABORTCALLED	Halt with processing halt request	ELIB_SDFS_RETERR_MEMFULL	Memory filled	ELIB_SDFS_RETERR_MISC	Others
Symbol	Type																
ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error																
ELIB_SDFS_RETERR_FILENUMMAX	Maximum directory or file number reached																
ELIB_SDFS_RETERR_IO	I-O error																
ELIB_SDFS_RETERR_ABORTCALLED	Halt with processing halt request																
ELIB_SDFS_RETERR_MEMFULL	Memory filled																
ELIB_SDFS_RETERR_MISC	Others																
Include file	srv_sdfs.h																
Calling sequence	int32_t Elib_SDFS_SaveToDirAsync(UINT Ap_ID, uint16_t DirType, uint8_t *NewTitleName);																
Argument	Type	I/O	Description														
Ap_ID	UINT	I	Application ID														
DirType	uint16_t	I	ELIB_SDFS_VIDEO: Dynamic image ELIB_SDFS_PICTURE: Static image														
NewTitleName	uint8_t *	I	This argument specifies a title name for a folder to be created. The caller should allocate the title field. Up to 64 bytes ending with null can be used.														

Return value	Type	-	Description
Ret	int32_t	O	<ul style="list-style-type: none"> - For normal end ELIB_SDFS_RETOK: Normal end (processing start) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_WRITEPROTECT: Write protect error ELIB_SDFS_RETERR_ABORTCALLED: Halt with halt request
Remark			

10.15 Folder deletion (asynchronous)

Classification	SD management file service support function																														
Function	Folder deletion	Symbol	Elib_SDFS_DeleteDirAsync																												
Function overview	This function deletes a folder specified with argument DataID (asynchronously).																														
<div>- The target folder need be initialized (Elib_SDFS_DirInitAsync).</div> <div>Check that there is a file which can be listed under the target folder.</div> <div><div>- Specify a value obtained with the Elib_SDFS_GetInfoList function in DataID.</div><div>- DCIM, SD-VIDEO, and SD-PIM directories cannot be deleted with this function.</div><div>- An error is returned if a folder with a specified ID is not registered.</div><div>- This function operates asynchronously.</div><div>- No event occurs unless this function terminates normally (processing starts) (return value: ELIB_SDFS_RETOK).</div></div> <div>After terminating folder deletion, this function creates and sends events listed below (info, subinfo = additional information in event transmission data).</div> <table><tr><th>Event symbol</th><th>info (transmission data)</th><th>subinfo (transmission data)</th><th>Explanation</th></tr><tr><td>SDFSNotify_DELETE_DIR_END</td><td>SDFSNotify_DELETE_DIR_END</td><td>ELIB_SDFS_RETOK</td><td>Normal end</td></tr><tr><td>SDFSNotify_DELETE_DIR_END</td><td>SDFSNotify_DELETE_DIR_END</td><td>Error type (negative)</td><td>Abnormal end</td></tr></table> <div><div>- Error types are listed below.</div><table><tr><th>Symbol</th><th>Type</th></tr><tr><td>ELIB_SDFS_RETERR_NODIR</td><td>No folder</td></tr><tr><td>ELIB_SDFS_RETERR_ILLEGAL_PARAM</td><td>Parameter error</td></tr><tr><td>ELIB_SDFS_RETERR_IO</td><td>I-O error</td></tr><tr><td>ELIB_SDFS_RETERR_ABORTCALLED</td><td>Halt with halt request</td></tr><tr><td>ELIB_SDFS_RETERR_DELFILE</td><td>File that can be deleted is present.</td></tr><tr><td>ELIB_SDFS_RETERR_NODELFILE</td><td>File that cannot be deleted is present.</td></tr><tr><td>ELIB_SDFS_RETERR_MISC</td><td>Others</td></tr></table></div>				Event symbol	info (transmission data)	subinfo (transmission data)	Explanation	SDFSNotify_DELETE_DIR_END	SDFSNotify_DELETE_DIR_END	ELIB_SDFS_RETOK	Normal end	SDFSNotify_DELETE_DIR_END	SDFSNotify_DELETE_DIR_END	Error type (negative)	Abnormal end	Symbol	Type	ELIB_SDFS_RETERR_NODIR	No folder	ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error	ELIB_SDFS_RETERR_IO	I-O error	ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request	ELIB_SDFS_RETERR_DELFILE	File that can be deleted is present.	ELIB_SDFS_RETERR_NODELFILE	File that cannot be deleted is present.	ELIB_SDFS_RETERR_MISC	Others
Event symbol	info (transmission data)	subinfo (transmission data)	Explanation																												
SDFSNotify_DELETE_DIR_END	SDFSNotify_DELETE_DIR_END	ELIB_SDFS_RETOK	Normal end																												
SDFSNotify_DELETE_DIR_END	SDFSNotify_DELETE_DIR_END	Error type (negative)	Abnormal end																												
Symbol	Type																														
ELIB_SDFS_RETERR_NODIR	No folder																														
ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error																														
ELIB_SDFS_RETERR_IO	I-O error																														
ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request																														
ELIB_SDFS_RETERR_DELFILE	File that can be deleted is present.																														
ELIB_SDFS_RETERR_NODELFILE	File that cannot be deleted is present.																														
ELIB_SDFS_RETERR_MISC	Others																														
Include file	srv_sdfs.h																														
Calling sequence	int32_t Elib_SDFS_DeleteDirAsync(UINT Ap_ID, uint16_t DirType, uint16_t DataID);																														
Argument	Type	I/O	Description																												
Ap_ID	UINT	I	Application ID																												
DirType	uint16_t	I	ELIB_SDFS_VIDEO: Dynamic image ELIB_SDFS_PICTURE: Static image																												
DataID	uint16_t	I	This argument specifies a data ID. Deletion: Set a number (1 to n)																												

Return value	Type	-	Description
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK: Normal end (processing start) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NODIR: Folder not found ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_WRITEPROTECT: Write protect error ELIB_SDFS_RETERR_NOTINIT: Folder not initialized
Remark			

10.16 File copy (asynchronous)

Classification	SD management file service support function																				
Function	File copy	Symbol	Elib_SDFS_CopyFileAsync																		
Function overview	This function copies a file specified with argument DataID (asynchronously).																				
<ul style="list-style-type: none">- Specify a value obtained with the Elib_SDFS_GetInfoList function in DataID.- An error is returned if a file with a specified ID is not registered.- This function does not support PIM data.- The copy destination folder need be initialized (Elib_SDFS_DirInitAsync).- This function operates asynchronously.- No event occurs unless this function terminates normally (processing starts) (return value: ELIB_SDFS_RETOK).- Set the pointer to an array storing the specified ID of a copy source in a shared memory. This shared memory is released by the SD file management.																					
After terminating file copying, this function creates and sends events listed below (info, subinfo = information in event transmission data).																					
<table><tr><th>Event symbol</th><th>info (transmission data)</th><th>subinfo (transmission data)</th><th>Explanation</th></tr><tr><td>SDFSNotify_COPY_END</td><td>SDFSNotify_COPY_END</td><td>ELIB_SDFS_RETOK</td><td>Normal end</td></tr><tr><td>SDFSNotify_COPY_END</td><td>SDFSNotify_COPY_END</td><td>Error type (negative)</td><td>Abnormal end</td></tr></table>				Event symbol	info (transmission data)	subinfo (transmission data)	Explanation	SDFSNotify_COPY_END	SDFSNotify_COPY_END	ELIB_SDFS_RETOK	Normal end	SDFSNotify_COPY_END	SDFSNotify_COPY_END	Error type (negative)	Abnormal end						
Event symbol	info (transmission data)	subinfo (transmission data)	Explanation																		
SDFSNotify_COPY_END	SDFSNotify_COPY_END	ELIB_SDFS_RETOK	Normal end																		
SDFSNotify_COPY_END	SDFSNotify_COPY_END	Error type (negative)	Abnormal end																		
<ul style="list-style-type: none">- Error types are listed below.																					
<table><tr><th>Symbol</th><th>Type</th></tr><tr><td>ELIB_SDFS_RETERR_NODIR</td><td>No folder</td></tr><tr><td>ELIB_SDFS_RETERR_NOFILE</td><td>No file</td></tr><tr><td>ELIB_SDFS_RETERR_ILLEGAL_PARAM</td><td>Parameter error</td></tr><tr><td>ELIB_SDFS_RETERR_FILENUMMAX</td><td>Maximum directory or file number reached</td></tr><tr><td>ELIB_SDFS_RETERR_IO</td><td>I-O error</td></tr><tr><td>ELIB_SDFS_RETERR_ABORTCALLED</td><td>Halt with halt request</td></tr><tr><td>ELIB_SDFS_RETERR_MEMFULL</td><td>Memory filled</td></tr><tr><td>ELIB_SDFS_RETERR_MISC</td><td>Others</td></tr></table>				Symbol	Type	ELIB_SDFS_RETERR_NODIR	No folder	ELIB_SDFS_RETERR_NOFILE	No file	ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error	ELIB_SDFS_RETERR_FILENUMMAX	Maximum directory or file number reached	ELIB_SDFS_RETERR_IO	I-O error	ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request	ELIB_SDFS_RETERR_MEMFULL	Memory filled	ELIB_SDFS_RETERR_MISC	Others
Symbol	Type																				
ELIB_SDFS_RETERR_NODIR	No folder																				
ELIB_SDFS_RETERR_NOFILE	No file																				
ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error																				
ELIB_SDFS_RETERR_FILENUMMAX	Maximum directory or file number reached																				
ELIB_SDFS_RETERR_IO	I-O error																				
ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request																				
ELIB_SDFS_RETERR_MEMFULL	Memory filled																				
ELIB_SDFS_RETERR_MISC	Others																				
The data below is passed as structure of data sent via the event transmitter (see Section 4.4).																					
uint16_t Number /* number of copied files */																					
Include file	srv_sdfs.h																				
Calling sequence	int32_t Elib_SDFS_CopyFileAsync(UINT Ap_ID, uint16_t FileType , int DataID_shmid ,uint32_t DataID_size,uint16_t CopyNumber, uint16_t DirDataID);																				
Argument	Type	I/O	Description																		

Ap_ID	UINT	I	Application ID
FileType	uint16_t	I	ELIB_SDFS_VIDEO: SD save dynamic image ELIB_SDFS_PICTURE: SD save static image
DataID_shmid	int	I	This argument specifies ID of a shared memory storing the copy source data ID array pointer.
DataID_size	uint32_t	I	This argument specifies the size of the copy source data ID array pointer.
CopyNumber	uint16_t	I	This argument specifies the number of files to be copied.
DirDataID	uint16_t	I	This argument specifies copy destination (folder) data ID.
Return value	Type	-	Description
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK: Normal end (processing start) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NODIR: Folder not found ELIB_SDFS_RETERR_NOFILE: File not found ELIB_SDFS_RETERR_NOTINIT:: Copy destination uninitialized ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_WRITEPROTECT: Write protect error
Remark			

10.17 File movement (asynchronous)

Classification	SD management file service support function		
Function	File movement	Symbol	Elib_SDFS_MoveFileAsync
Function overview	<p>This function moves a file to a specified folder (asynchronously).</p> <ul style="list-style-type: none"> - Specify a value obtained with the Elib_SDFS_GetInfoList function in DataID. - An error is returned if a file with a specified ID is not registered. - This function does not support PIM data. - The move destination folder need be initialized (Elib_SDFS_DirInitAsync). - This function operates asynchronously. - Set the pointer to an array storing the specified ID of a target file in a shared memory. This shared memory is released by the SD file management. <p>After terminating file movement, this function creates and sends events listed below (info, subinfo = information in event transmission data).</p>		

Event symbol	info (transmission data)	subinfo (transmission data)	Explanation
SDFSNotify_MOVE_END	SDFSNotify_MOVE_END	ELIB_SDFS_RETOK	Normal end
SDFSNotify_MOVE_END	SDFSNotify_MOVE_END	Error type (negative)	Abnormal end

- Error types are listed below.

Symbol	Type
ELIB_SDFS_RETERR_NODIR	No folder
ELIB_SDFS_RETERR_NOFILE	No file
ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error
ELIB_SDFS_RETERR_FILENUMMAX	Maximum directory or file number reached
ELIB_SDFS_RETERR_IO	I-O error
ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request
ELIB_SDFS_RETERR_MEMFULL	Memory filled
ELIB_SDFS_RETERR_MISC	Others

The data below is passed as structure of data sent via the event transmitter (see Section 4.4).

uint16_t Number /* number of moved files */

Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_MoveFileAsync(UINT Ap_ID, uint16_t FileType, int DataID_shmid, uint32_t DataID__size, uint16_t MoveNumber, uint16_t DirDataID);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
FileType	uint16_t	I	ELIB_SDFS_VIDEO: SD save dynamic image ELIB_SDFS_PICTURE: SD save static image
DataID_shmid	int	I	This argument specifies ID of a shared memory storing the move source data ID array pointer.
DataID_size	uint32_t	I	This argument specifies the size of the move source data ID array pointer.
MoveNumber	uint16_t	I	This argument specifies the number of files to be moved.
DirDataID	uint16_t	I	This argument specifies move destination (folder) data ID.
Return value	Type	-	Description
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK: Normal end (processing start) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NODIR: Folder not found ELIB_SDFS_RETERR_NOFILE: File not found ELIB_SDFS_RETERR_NOTINIT: Move destination uninitialized ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors

			ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_WRITEPROTECT: Write protect error
Remark			

10.18 Title change (asynchronous)

Classification

Function

Function overview

SD management file service support function

Title change

Symbol

Elib_SDFS_ChgTitleAsync

This function changes the title of data specified with DataID.

- Specify a value obtained with the Elib_SDFS_GetInfoList function in DataID.

- The caller should allocate the title field.

- Up to 64 bytes ending with null can be used.

- An error is returned if a file with a specified ID is not registered.

- This function operates asynchronously.

- No event occurs unless this function terminates normally (processing starts) (return value: ELIB_SDFS_RETOK).

- When a folder is a target, it need be initialized (Elib_SDFS_DirInitAsync).

When a file is a target, its parent folder need be initialized.

After terminating title change, this function creates and sends events listed below (info, subinfo = additional information in event transmission data).

Event symbol	info (transmission data)	subinfo (transmission data)	Explanation
SDFSNotify_TITLECHG_END	SDFSNotify_TITLECHG_END	ELIB_SDFS_RETOK	Normal end
SDFSNotify_TITLECHG_END	SDFSNotify_TITLECHG_END	Error type (negative)	Abnormal end

- Error types are listed below.

Symbol	Type
ELIB_SDFS_RETERR_NOFILE	No file
ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error
ELIB_SDFS_RETERR_IO	I-O error

ELIB_SDFS_RETERR_ABORTCALLED		Halt with halt request	
ELIB_SDFS_RETERR_MEMFULL		Memory filled	
ELIB_SDFS_RETERR_NOTINIT		Copy destination uninitialized	
ELIB_SDFS_RETERR_MISC		Others	

Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_ChgTitleAsync(UINT Ap_ID, uint16_t DataID, uint16_t DirFileType, uint8_t *NewTitleName);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
DataID	uint16_t	I	This argument specifies data ID (folder or file).
DirFileType	uint16_t	I	ELIB_SDFS_DIR: Flder ELIB_SDFS_VIDEO: Dynamic image ELIB_SDFS_PICTURE: Static image ELIB_SDFS_PIM:PIM
NewTitleName	uint8_t *	I	This argument specifies the pointer to a title storage field. The caller should allocate the title storage field. Up to 64 bytes ending with null can be used.
Return value	Type	-	Description
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK: Normal end (processing start) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NODIR: Folder not found ELIB_SDFS_RETERR_NOFILE: File not found ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_WRITEPROTECT: Write protect error
Remark			

10.19 Single vObject export (asynchronous)

Classification	SD management file service support function																	
Function	Single vObject export	Symbol	Elib_SDFS_PIM_ExportAsync															
Function overview	This function saves specified vObject in an SD card.																	
<div>- This function operates asynchronously.</div> <div>- T_SRV_SDPIIME_REQ_EXPORT members are listed below.</div> <div>- No event occurs unless this function terminates normally (processing starts) (return value: ELIB_SDFS_RETOK).</div>																		
<table><tr><td>Variable</td><td>I/O</td><td>Explanation</td></tr><tr><td>unsigned char vobjinfo</td><td>I</td><td>Types of obtained vObject - ELIB_SDFS_CARD vCard - ELIB_SDFS_EVENT vEvent - ELIB_SDFS_TODO vTODO - ELIB_SDFS_INBOX Received mail - ELIB_SDFS_OUTBOX Untransmitted mail - ELIB_SDFS_SENTBOX Transmitted mail - ELIB_SDFS_NOTE Free memorandum - ELIB_SDFS_BOOKMARK Bookmark - ELIB_SDFS_MYPB Personal data (name, telephone number, e-mail address) - ELIB_SDFS_MYPB_ALL Personal data (all)</td></tr><tr><td>unsigned char dial_len</td><td>I</td><td>Local office number size (Set a dial_data size. Its must not exceed ELIB_MSISDN_DIAL_MAX. If dial_data is null, set 0.)</td></tr><tr><td>unsigned short rec</td><td>I</td><td>Access number for data in mobile phone (vObject ELIB_SDFS_MYPB or ELIB_SDFS_MYPB_ALL is obtained, specify ELIB_SDFS_INVALIDREC.)</td></tr><tr><td>unsigned char* dial_data</td><td>I</td><td>Local office number (valid when vObject ELIB_SDFS_MYPB or ELIB_SDFS_MYPB_ALL is obtained. For other vObject types, specify null.) (Array “unsigned char[ELIB_MSISDN_DIAL_MAX]” is set to specify the starting address in the destination.)</td></tr></table>				Variable	I/O	Explanation	unsigned char vobjinfo	I	Types of obtained vObject - ELIB_SDFS_CARD vCard - ELIB_SDFS_EVENT vEvent - ELIB_SDFS_TODO vTODO - ELIB_SDFS_INBOX Received mail - ELIB_SDFS_OUTBOX Untransmitted mail - ELIB_SDFS_SENTBOX Transmitted mail - ELIB_SDFS_NOTE Free memorandum - ELIB_SDFS_BOOKMARK Bookmark - ELIB_SDFS_MYPB Personal data (name, telephone number, e-mail address) - ELIB_SDFS_MYPB_ALL Personal data (all)	unsigned char dial_len	I	Local office number size (Set a dial_data size. Its must not exceed ELIB_MSISDN_DIAL_MAX. If dial_data is null, set 0.)	unsigned short rec	I	Access number for data in mobile phone (vObject ELIB_SDFS_MYPB or ELIB_SDFS_MYPB_ALL is obtained, specify ELIB_SDFS_INVALIDREC.)	unsigned char* dial_data	I	Local office number (valid when vObject ELIB_SDFS_MYPB or ELIB_SDFS_MYPB_ALL is obtained. For other vObject types, specify null.) (Array “unsigned char[ELIB_MSISDN_DIAL_MAX]” is set to specify the starting address in the destination.)
Variable	I/O	Explanation																
unsigned char vobjinfo	I	Types of obtained vObject - ELIB_SDFS_CARD vCard - ELIB_SDFS_EVENT vEvent - ELIB_SDFS_TODO vTODO - ELIB_SDFS_INBOX Received mail - ELIB_SDFS_OUTBOX Untransmitted mail - ELIB_SDFS_SENTBOX Transmitted mail - ELIB_SDFS_NOTE Free memorandum - ELIB_SDFS_BOOKMARK Bookmark - ELIB_SDFS_MYPB Personal data (name, telephone number, e-mail address) - ELIB_SDFS_MYPB_ALL Personal data (all)																
unsigned char dial_len	I	Local office number size (Set a dial_data size. Its must not exceed ELIB_MSISDN_DIAL_MAX. If dial_data is null, set 0.)																
unsigned short rec	I	Access number for data in mobile phone (vObject ELIB_SDFS_MYPB or ELIB_SDFS_MYPB_ALL is obtained, specify ELIB_SDFS_INVALIDREC.)																
unsigned char* dial_data	I	Local office number (valid when vObject ELIB_SDFS_MYPB or ELIB_SDFS_MYPB_ALL is obtained. For other vObject types, specify null.) (Array “unsigned char[ELIB_MSISDN_DIAL_MAX]” is set to specify the starting address in the destination.)																
After terminating export, this function creates and sends events listed below (info, subinfo = additional information in event transmission data).																		
<table><tr><td>Event symbol</td><td>info (transmission data)</td><td>subinfo (transmission data)</td><td>Explanation</td></tr><tr><td>SDFSNotify_PIMBACKUP_END</td><td>SDFSNotify_PIMBACKUP_END</td><td>ELIB_SDFS_RETOK</td><td>Normal end</td></tr><tr><td>SDFSNotify_PIMBACKUP_END</td><td>SDFSNotify_PIMBACKUP_END</td><td>Error type (negative)</td><td>Abnormal end</td></tr></table>				Event symbol	info (transmission data)	subinfo (transmission data)	Explanation	SDFSNotify_PIMBACKUP_END	SDFSNotify_PIMBACKUP_END	ELIB_SDFS_RETOK	Normal end	SDFSNotify_PIMBACKUP_END	SDFSNotify_PIMBACKUP_END	Error type (negative)	Abnormal end			
Event symbol	info (transmission data)	subinfo (transmission data)	Explanation															
SDFSNotify_PIMBACKUP_END	SDFSNotify_PIMBACKUP_END	ELIB_SDFS_RETOK	Normal end															
SDFSNotify_PIMBACKUP_END	SDFSNotify_PIMBACKUP_END	Error type (negative)	Abnormal end															

- Error types are listed below.

Symbol	Type
ELIB_SDFS_RETERR_NOFILE	No file
ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error
ELIB_SDFS_RETERR_IO	I-O error
ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request
ELIB_SDFS_RETERR_FILENUMMAX	Maximum directory or file number reached
ELIB_SDFS_RETERR_MEMFULL	Memory filled
ELIB_SDFS_RETERR_MISC	Others

Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_PIM_ExportAsync(UINT Ap_ID, T_SRV_SDPIME_REQ_EXPORT *t_export);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
t_export	T_SRV_SDPIME_REQ_EXPORT*	I	This argument specifies the pointer to structure to be exported.
Return value	Type	-	Description
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK: Normal end (processing start) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_WRITEPROTECT: Write protect error
Remark			

10.20 All vObject items export (asynchronous)

Classification	SD management file service support function		
Function	All vObject items export	Symbol	Elib_SDFS_PIM_ExportAllAsync
Function overview	This function saves all of specified PIM data items in an SD card.		

- This function operates asynchronously.
- T_SRV_SDPIIME_REQ_EXPORTALL members are listed below.
- No event occurs unless this function terminates normally (processing starts) (return value: ELIB_SDFS_RETOK).

Variable	I/O	Explanation
Unsigned char vobjinfo	I	Types of obtained vObject - ELIB_SDFS_CARD vCard - ELIB_SDFS_EVENT vEvent - ELIB_SDFS_TODO vTODO - ELIB_SDFS_EVT0 vEvent/vTODO - ELIB_SDFS_INBOX Received mail - ELIB_SDFS_OUTBOX Untransmitted mail - ELIB_SDFS_SENTBOX Transmitted mail - ELIB_SDFS_NOTE Free memorandum - ELIB_SDFS_BOOKMARK Bookmark
unsigned char dial_len	I	Local office number size (Set a dial_data size. Its must not exceed ELIB_MSISDN_DIAL_MAX. If dial_data is null, set 0.)
unsigned char reserved[2]	-	Boundary
unsigned char* dial_data	I	Local office number (valid when vObject ELIB_SDFS_CARD is obtained. For other vObject types, specify null.) (Array “unsigned char[ELIB_MSISDN_DIAL_MAX]” is set to specify the starting address in the destination.)

After terminating export, this function creates and sends events listed below (info, subinfo = additional information in event transmission data).

Event symbol	info (transmission data)	subinfo (transmission data)	Explanation
SDFSNotify_ALL_PIMBACKUP_END	SDFSNotify_ALL_PIMBACKUP_END	ELIB_SDFS_RETOK	Normal end
SDFSNotify_ALL_PIMBACKUP_END	SDFSNotify_ALL_PIMBACKUP_END	Error type (negative)	Abnormal end

- Error types are listed below.

Symbol	Type
ELIB_SDFS_RETERR_NOFILE	No file
ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error
ELIB_SDFS_RETERR_IO	I-O error

ELIB_SDFS_RETERR_ABORTCALLED		Halt with halt request	
ELIB_SDFS_RETERR_FILENUMMAX		Maximum directory or file number reached	
ELIB_SDFS_RETERR_MEMFULL		Memory filled	
ELIB_SDFS_RETERR_MISC		Others	

Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_PIM_ExportAllAsync(UINT Ap_ID, T_SRV_SDPIME_REQ_EXPORTALL *t_exportall);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
t_exportall	T_SRV_SDPIME_REQ_EXPORTALL *	I	This argument specifies the pointer to structure to be exported.
Return value	Type	-	Description
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK: Normal end (processing start) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_WRITEPROTECT: Write protect error
Remark			

10.21 Single vObject import (asynchronous)

Classification	SD management file service support function		
Function	Single vObject import	Symbol	Elib_SDFS_PIM_ImportAsync
Function overview	This function saves a vObject item with a specified data ID in a terminal.		

- This function operates asynchronously.
- T_SRV_SDPIIME_REQ_IMPORT members are listed below.

Variable	I/O	Explanation
unsigned char vobjinfo	I	Types of obtained vObject - ELIB_SDFS_CARD vCard - ELIB_SDFS_EVENT Event - ELIB_SDFS_TODO vTODO - ELIB_SDFS_EMAIL_IN eceived e-mail - ELIB_SDFS_EMAIL_OUT Untransmitted e-mail - ELIB_SDFS_EMAIL_SENT ransmitted e-mail - ELIB_SDFS_SMS_IN Received SMS-mail - ELIB_SDFS_SMS_OUT ntransmitted SMS-mail - ELIB_SDFS_SMS_SENT ransmitted SMS-mail - ELIB_SDFS_NOTE ree memorandum - ELIB_SDFS_BOOKMARK Bookmark
unsigned char reserve	-	Boundary
unsigned short id	I	ID of file to be read
signed long pos	I	File read position. For object selection, set a position obtained with vObject list information acquisition function as an address. For file selection, always set 0.
signed long vobjsize	I	Obtained vObject size (vobjsize obtained with the vObject list information acquisition function)

After terminating import, this function creates and sends events listed below (info, subinfo = additional information in event transmission data).

Event symbol	info (transmission data)	subinfo (transmission data)	Explanation
SDFSNotify_PIMIMPORT_END	SDFSNotify_PIMIMPORT_END	ELIB_SDFS_RETOK	Normal end
SDFSNotify_PIMIMPORT_END	SDFSNotify_PIMIMPORT_END	Error type (negative)	Abnormal end

- Error types are listed below.

Symbol	Type
ELIB_SDFS_RETERR_NOFILE	No directory
ELIB_SDFS_RETERR_ILLEGAL_PARAMETER	Parameter error
ELIB_SDFS_RETERR_IO	I-O error
ELIB_SDFS_RETERR_BLOCKING	Blocking error (during SD management file service processing)
ELIB_SDFS_RETERR_ABORTCALLED	Halt with processing halt request
ELIB_SDFS_RETERR_FILENUMMAX	Maximum directory or file number reached
ELIB_SDFS_RETERR_MEMFULL	Memory filled

ELIB_SDFS_RETERR_MISC		Others	
Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_PIM_ImportAsync(UINT Ap_ID, T_SRV_SDPIME_REQ_IMPORT *t_import);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
t_import	T_SRV_SDPIME_REQ_IMPORT *	I	This argument specifies the pointer to structure to be imported.
Return value	Type	-	Description
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK: Normal end (processing start) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_WRITEPROTECT: Write protect error
Remark			

10.22 All vObject items import (asynchronous)

Classification	SD management file service support function		
Function	All vObject items import	Symbol	Elib_SDFS_PIM_ImportAll Async
Function overview	This function saves all of vObject items with a specified data ID in a terminal.		

- This function operates asynchronously.
- T_SRV_SDPIIME_REQ_IMPORTALL members are listed below.
- No event occurs unless this function terminates normally (processing starts) (return value: ELIB_SDFS_RETOK).

Variable	I/O	Rxplanation
unsigned char vobjinfo	I	Types of obtained vObject - ELIB_SDFS_CARD vCard - ELIB_SDFS_EVENT vEvent - ELIB_SDFS_TODO vTODO - ELIB_SDFS_EVT0 vEvent/vTODO - ELIB_SDFS_INBOX Received mail - ELIB_SDFS_OUTBOX Untransmitted mail - ELIB_SDFS_SENTBOX Transmitted mail - ELIB_SDFS_NOTE Free memorandum - ELIB_SDFS_BOOKMARK Bookmark
unsigned char mode	I	Import type (Specify overwrite or addition as follows:) - ELIB_SDFS_PIM_OVERWRITE: Overwrite - ELIB_SDFS_PIM_ADDITION: Addition
unsigned short id	I	Read file ID
unsigned char mypb	I	Whether local office number is imported (valid only when vCard is obtained) - ELIB_SDFS_SDPIM_IMP_MYPB Imported - ELIB_SDFS_SDPIM_IMP_NON Not imported
unsigned char reserved[3]	-	Boundary

After terminating import, this function creates and sends events listed below (info, subinfo = additional information in event transmission data).

Event symbol	info (transmission data)	subinfo (transmission data)	Explanation
SDFSNotify_ALL_PIMIMPOR T_END	SDFSNotify_ALL_PIMIMPORT_E ND	ELIB_SDFS_RETOK	Normal end
SDFSNotify_ALL_PIMIMPOR T_END	SDFSNotify_ALL_PIMIMPORT_E ND	Error type (negative)	Abnormal end

- Error types are listed below.

Symbol	Type
--------	------

ELIB_SDFS_RETERR_NOFILE		No file	
ELIB_SDFS_RETERR_ILLEGAL_PARAM		Parameter error	
ELIB_SDFS_RETERR_IO		I-O error	
ELIB_SDFS_RETERR_ABORTCALLED		Halt with processing halt request	
ELIB_SDFS_RETERR_FILENUMMAX		Maximum directory or file number reached	
ELIB_SDFS_RETERR_MEMFULL		Memory filled	
ELIB_SDFS_RETERR_MISC		Others	

Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_PIM_ImportAllAsync(UINT Ap_ID, T_SRV_SDPIME_REQ_IMPORTALL *t_importall);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
t_importall	T_SRV_SDPIME_REQ_IMPORTALL *	I	This argument specifies the pointer to structure to be imported.
Return value	Type	-	Description
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK: Normal end (processing start) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_ILLEGAL_PA RAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_WRITEPROT ECT: Write protect error
Remark			

10.23 Folder initialization (asynchronous)

Classification	SD management file service support function																										
Function	Folder initialization	Symbol	ELib_SDFS_DirInitAsync																								
Function overview	This function creates management information in a folder.																										
<div>- SD management information is necessary to execute processing related to SD management file service.</div> <div>This function creates SD management information and initializes the target folder.</div> <div>- This function operates asynchronously.</div> <div>If a target folder has been initialized, this fact can be recognized with an error value but an event also occurs.</div> <div>- For PIM data, the caller is recommended to display a pop-up screen reporting that synchronous operation is under process because SD management information is not created but asynchronous operation is performed.</div> <div>- No event occurs unless this function terminates normally or the target folder has been initialized.</div> <div>After terminating folder initialization, this function creates and sends events listed below (info, subinfo = information in event transmission data)</div> <table><tr><th>Event symbol</th><th>info (transmission data)</th><th>subinfo (transmission data)</th><th>Explanation</th></tr><tr><td>SDFSNotify_INIT_END</td><td>SDFSNotify_INIT_END</td><td>ELIB_SDFS_RETOK ELIB_SDFS_RET_INITEND</td><td>Normal end</td></tr><tr><td>SDFSNotify_INIT_END</td><td>SDFSNotify_INIT_END</td><td>Error type (negative)</td><td>Abnormal end</td></tr></table> <div>- Error types are listed below.</div> <table><tr><th>Symbol</th><th>Type</th></tr><tr><td>ELIB_SDFS_RETERR_NODIR</td><td>No directory</td></tr><tr><td>ELIB_SDFS_RETERR_ILLEGAL_PARAM</td><td>Parameter error</td></tr><tr><td>ELIB_SDFS_RETERR_IO</td><td>I-O error</td></tr><tr><td>ELIB_SDFS_RETERR_ABORTCALLED</td><td>Halt with halt request</td></tr><tr><td>ELIB_SDFS_RETERR_MISC</td><td>Others</td></tr></table>				Event symbol	info (transmission data)	subinfo (transmission data)	Explanation	SDFSNotify_INIT_END	SDFSNotify_INIT_END	ELIB_SDFS_RETOK ELIB_SDFS_RET_INITEND	Normal end	SDFSNotify_INIT_END	SDFSNotify_INIT_END	Error type (negative)	Abnormal end	Symbol	Type	ELIB_SDFS_RETERR_NODIR	No directory	ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error	ELIB_SDFS_RETERR_IO	I-O error	ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request	ELIB_SDFS_RETERR_MISC	Others
Event symbol	info (transmission data)	subinfo (transmission data)	Explanation																								
SDFSNotify_INIT_END	SDFSNotify_INIT_END	ELIB_SDFS_RETOK ELIB_SDFS_RET_INITEND	Normal end																								
SDFSNotify_INIT_END	SDFSNotify_INIT_END	Error type (negative)	Abnormal end																								
Symbol	Type																										
ELIB_SDFS_RETERR_NODIR	No directory																										
ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error																										
ELIB_SDFS_RETERR_IO	I-O error																										
ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request																										
ELIB_SDFS_RETERR_MISC	Others																										
Include file	srv_sdifs.h																										
Calling sequence	int32_t Elib_SDFS_DirInitAsync(UINT Ap_ID, uint16_t DataID);																										
Argument	Type	I/O	Description																								
Ap_ID	UINT	I	Application ID																								
DataID	uint16_t	I	This argument specifies data ID of a folder to be initialized.																								
Return value	Type	-	Description																								

Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK: Normal end (request accepted) ELIB_SDFS_RET_INITEND: Target folder initialized - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NODIR: No folder found ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_ABORTCALLED: Halt with halt request
Remark			

10.24 Total folder/file list count acquisition

Classification	SD management file service support function		
Function	Total folder/file list count acquisition	Symbol	Elib_SDFS_GetListCount
Function overview	<p>This function obtains the total number of file lists of a specified type in a folder.</p> <p>•Folder type symbol ELIB_SDFS_DIR</p> <p>•File type symbol</p> <ul style="list-style-type: none"> - Basic file type <ul style="list-style-type: none"> ELIB_SDFS_PICTURE : Static image stored in SD card ELIB_SDFS_VIDEO : Dynamic image stored in SD card ELIB_SDFS_PIM : PIM stored in SD card - Individual file type (static image/dynamic image) <ul style="list-style-type: none"> ELIB_SDFS_PICTURE_JPEG : JPEG stored in SD card ELIB_SDFS_VIDEO_ASF : ASF dynamic image stored in SD card ELIB_SDFS_VIDEO_3GP : 3GP dynamic image stored in SD card ELIB_SDFS_VIDEO_SDV : SDV dynamic image stored in SD card ELIB_SDFS_VIDEO_MP4 : MP4 dynamic image stored in SD card - Individual file type (PIM) <ul style="list-style-type: none"> ELIB_SDFS_PIM_CARD:vCard ELIB_SDFS_PIM_CALENDAR:vCalendar ELIB_SDFS_PIM_INBOX: Received mail ELIB_SDFS_PIM_OUTBOX: Untransmitted mail ELIB_SDFS_PIM_SENTBOX: Transmitted mail ELIB_SDFS_PIM_NOTE: Free memorandum ELIB_SDFS_PIM_BOOKMARK: Bookmark - Disjunction (or operation value) of the above folder type symbol and file type symbol can be specified as a folder/file type. Example) Specify the following for the number of MP4 files (dynamic images) under a folder with a specified data ID ELIB_SDFS_DIR ELIB_SDFS_VIDEO_MP4 Note that disjunction of file types (individual file basic file and individual file individual file, for example) cannot be specified - A specified folder need be initialized (Elib_SDFS_DirInitAsync) when static or dynamic image data is target.. The folder need not be initialized when PIM data is target. 		
Include file	srv_sdfs.h		
Calling sequence	<pre>int32_t Elib_SDFS_GetListCount(UINT Ap_ID, uint16_t DataID, uint16_t *DirNumber, uint16_t *FileNumber, uint16_t DirFileType);</pre>		

Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
DataID	uint16_t	I	<p>This argument specifies a folder data ID as follows:</p> <p>Folder under a directory in hierarchy not higher than that of DCIM directory:ELIB_SDFS_ROOTDIR_DCIM</p> <p>Folder under a directory in hierarchy not higher than that of SD-VIDEO directory:ELIB_SDFS_ROOTDIR_SDVIDEO</p> <p>Telephone directory root folder data ID (PIM):ELIB_SDFS_ROOTDIR_CARD</p> <p>Schedule root folder data ID (PIM):ELIB_SDFS_ROOTDIR_CALENDAR</p> <p>Received mail root folder data ID (PIM):ELIB_SDFS_ROOTDIR_INBOX</p> <p>Untransmitted mail root folder data ID (PIM):ELIB_SDFS_ROOTDIR_OUTBOX</p> <p>Transmitted mail root folder data ID (PIM):ELIB_SDFS_ROOTDIR_SENTBOX</p> <p>Free memorandum root folder data ID (PIM):ELIB_SDFS_ROOTDIR_NOTE</p> <p>Bookmark root folder data ID (PIM):ELIB_SDFS_ROOTDIR_BOOKMARK</p> <p>All PIM folders under a directory in hierarchy not higher than SD card:ELIB_SDFS_ROOTDIR_SDPIM</p>
DirNumber	uint16_t*	I	This argument specifies the number of folders (the caller should allocate the field).
FileNumber	uint16_t*	I	This argument specifies the number of files (the caller should allocate the field).
DirFileType	uint16_t	I	<p>This argument specifies a folder/file type.</p> <p>Disjunction of the folder type symbol below and one of file type symbol listed below can be specified as a folder/file type.</p> <p><u><Folder type symbol></u> ELIB_SDFS_DIR</p> <p><u><File type symbol></u> ELIB_SDFS_PICTURE General static images stored in SD card ELIB_SDFS_PICTURE_JPEG JPEG stored in SD card</p> <p>ELIB_SDFS_VIDEO General dynamic images stored in SD card ELIB_SDFS_VIDEO_ASF ASF dynamic image stored in SD card ELIB_SDFS_VIDEO_3GP 3GP dynamic image stored in</p>

			SD card ELIB_SDFS_VIDEO_SDV : SDV dynamic image stored in SD card ELIB_SDFS_VIDEO_MP4 : MP4 dynamic image stored in SD card ELIB_SDFS_PIM : PIM stored in SD card ELIB_SDFS_PIM_CARD:vCard ELIB_SDFS_PIM_CALENDAR:vCalendar ELIB_SDFS_PIM_INBOX: Received mail ELIB_SDFS_PIM_OUTBOX: Untransmitted mail ELIB_SDFS_PIM_SENTBOX: Transmitted mail ELIB_SDFS_PIM_NOTE: Free memorandum ELIB_SDFS_PIM_BOOKMARK: Bookmark
Return value	Type	-	Description
Ret	int32_t	O	For normal end ELIB_SDFS_RETOK: Normal end ELIB_SDFS_RET_INITEND: Target folder initialized - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NODIR: No folder found ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_BLOCKING: Blocking error ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_NOTINIT: Folder uninitialized ELIB_SDFS_RETERR_MISC: Other errors
Remark			

10.25 Folder/file list information acquisition

Classification	SD management file service support function		
Function	Folder/file information acquisition	Symbol	Elib_SDFS_GetInfoList
Function overview	This function obtains a detailed information list of files under a specified folder and folders lower hierarchically than it.		
<div><div><div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><</div></div></div></div>			

	ELIB_FA_READONLY: Read-only file ELIB_FA_HIDDEN: Hidden file ELIB_FA_SYSTEM: System file ELIB_FA_DIR: Subdirectory ELIB_FA_ARCH: Archive file		
uint8_t Multiflag	Whether folder/file with the same name is present ELIB_SDFS_MULTI_OFF: Not present ELIB_SDFS_MULTI_ON: Present		
uint8_t Dpoffflag	DPOF flag status ELIB_SDFS_DPOF_OFF: Cleared ELIB_SDFS_DPOF_ON: Set		
uint8_t dummy	Dummy		
Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_GetInfoList(UINT Ap_ID, uint16_t DataID, uint16_t DirFileType, uint16_t Namemode, uint16_t Direct, uint16_t GetStartNumber, uint16_t GetNumber, ELIB_SDFS_DIRFILE_INFO *List);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
DataID	Uint16_t	I	This argument specifies a folder data ID as follows: Folder under a directory in hierarchy not higher than that of DCIM directory:ELIB_SDFS_ROOTDIR_DCIM Folder under a directory in hierarchy not higher than that of SD-VIDEO directory:ELIB_SDFS_ROOTDIR_SDVIDEO Telephone directory root folder data ID (PIM):ELIB_SDFS_ROOTDIR_CARD Schedule root folder data ID (PIM):ELIB_SDFS_ROOTDIR_CALENDAR Received mail root folder data ID (PIM):ELIB_SDFS_ROOTDIR_INBOX Untransmitted mail root folder data ID (PIM):ELIB_SDFS_ROOTDIR_OUTBOX Transmitted mail root folder data ID (PIM):ELIB_SDFS_ROOTDIR_SENTBOX Free memorandum root folder data ID (PIM):ELIB_SDFS_ROOTDIR_NOTE Bookmark root folder data ID (PIM):ELIB_SDFS_ROOTDIR_BOOKMARK All PIM folders under a directory in hierarchy not higher than SD card:ELIB_SDFS_ROOTDIR_SDPIM
DirFileType	uint16_t	I	This argument specifies a folder as follows: ELIB_SDFS_DIR File type symbol

			ELIB_SDFS_PICTURE General static images stored in SD card ELIB_SDFS_PICTURE_JPEG JPEG stored in SD card ELIB_SDFS_VIDEO General dynamic images stored in SD card ELIB_SDFS_VIDEO_ASF ASF dynamic image stored in SD card ELIB_SDFS_VIDEO_3GP 3GP dynamic image stored in SD card ELIB_SDFS_VIDEO_SDV SDV dynamic image stored in SD card ELIB_SDFS_VIDEO_MP4 MP4 dynamic image stored in SD card ELIB_SDFS_PIM : PIM stored in SD card ELIB_SDFS_PIM_CARD:vCard ELIB_SDFS_PIM_CALENDAR:vCalendar ELIB_SDFS_PIM_INBOX: Received mail ELIB_SDFS_PIM_OUTBOX: Untransmitted mail ELIB_SDFS_PIM_SENTBOX: Transmitted mail ELIB_SDFS_PIM_NOTE: Free memorandum ELIB_SDFS_PIM_BOOKMARK: Bookmark
NameMode	uint16_t	I	This argument specifies name mode. Name for list display <Folder/file name> ELIB_SDFS_DIRFILE_NAME <Title> ELIB_SDFS_TITLE_NAME
Direct	uint16_t	I	This argument specifies an acquisition direction as follows: 0: Smallest No. to Largest No. 1: Largest No. to smallest No. For PIM, acquired order is specified automatically.
GetStartNumber	uint16_t	I	This argument specifies an acquisition start number: 1 (data displayed at the top) to n
GetNumber	uint16_t	I	This argument specifies the number of lists to be obtained.
List	ELIB_SDFS_DIRFILEINFO *	O	This argument indicates the pointer to the folder/file information structure buffer (The application side should allocate an area for the specified number of files.)
Return value	Type	-	Description

Ret	int32_t	O	- For normal end (0 or positive numeric) Unregistered :0 Number of obtained files:1ton (n<GetNumber) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NODIR: Folder not found ELIB_SDFS_RETERR_NOTINIT: Folder not initialized ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_RETERR_ABORTCALLED: Halt with halt request
Remarks			

10.26 Total free area size acquisition

Classification	SD management file service support function		
Function	Total free area size acquisition	Symbol	Elib_SDFS_GetFreeSize
Function overview	<p>This function returns the total free area size in an SD card by using the number of bytes.</p> <ul style="list-style-type: none"> - This function returns the total free area size in an SD card. 		
Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_GetFreeSize(UINT Ap_ID);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
Return value	Type	-	Description
Ret	int32_t	O	<ul style="list-style-type: none"> - For normal end (0 or negative numeric) 0<=Free area size (number of bytes) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_MISC: Other errors
Remark	<p>* This function does not support 2G bytes as a size to be returned.</p>		

10.27 SD card information acquisition

Classification	SD management file service support function																																																																																																	
Function	SD card information acquisition					Symbol	Elib_SDFS_GetCardInfo																																																																																											
Function overview	This function obtains information about an SD card and processing status of it.																																																																																																	
This function returns card information listed below with card ID (16-byte ID specific to a card).																																																																																																		
* A card ID can be used to check if a card inserted in the preceding processing is inserted again because a card ID is specific to a card).																																																																																																		
<table><tr><td>bit7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>Explanation</td><td colspan="2">Symbol</td></tr><tr><td>1</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>While inserted:ON</td><td colspan="2">ELIB_SDFS_INSERT_SD:0x80</td></tr><tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>1</td><td>If format unsupported:ON</td><td colspan="2">ELIB_SDFS_UNKNOWNFORMAT:0x01</td></tr><tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>1</td><td>X</td><td>If card size unsupported:ON</td><td colspan="2">ELIB_SDFS_UNKNOWNCARDSIZE:0x02</td></tr><tr><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>1</td><td>X</td><td>X</td><td>If card type unsupported:ON</td><td colspan="2">ELIB_SDFS_UNKNOWNCARDKIND:0x04</td></tr><tr><td>X</td><td>X</td><td>X</td><td>X</td><td>1</td><td>X</td><td>X</td><td>X</td><td>When write protected:ON</td><td colspan="2">ELIB_SDFS_READY_WRITEPROTECT:0x08</td></tr><tr><td>X</td><td>X</td><td>X</td><td>1</td><td>X</td><td>X</td><td>X</td><td>X</td><td>While card recognition under process:ON</td><td colspan="2">ELIB_SDFS_INITIAL_SDCARD:0x10</td></tr><tr><td>X</td><td>X</td><td>1</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>FAT entry: Error</td><td colspan="2">ELIB_SDFS_UNUSUAL_FATENTRY:0x20</td></tr></table>											bit7	6	5	4	3	2	1	0	Explanation	Symbol		1	X	X	X	X	X	X	X	While inserted:ON	ELIB_SDFS_INSERT_SD:0x80		X	X	X	X	X	X	X	1	If format unsupported:ON	ELIB_SDFS_UNKNOWNFORMAT:0x01		X	X	X	X	X	X	1	X	If card size unsupported:ON	ELIB_SDFS_UNKNOWNCARDSIZE:0x02		X	X	X	X	X	1	X	X	If card type unsupported:ON	ELIB_SDFS_UNKNOWNCARDKIND:0x04		X	X	X	X	1	X	X	X	When write protected:ON	ELIB_SDFS_READY_WRITEPROTECT:0x08		X	X	X	1	X	X	X	X	While card recognition under process:ON	ELIB_SDFS_INITIAL_SDCARD:0x10		X	X	1	X	X	X	X	X	FAT entry: Error	ELIB_SDFS_UNUSUAL_FATENTRY:0x20	
bit7	6	5	4	3	2	1	0	Explanation	Symbol																																																																																									
1	X	X	X	X	X	X	X	While inserted:ON	ELIB_SDFS_INSERT_SD:0x80																																																																																									
X	X	X	X	X	X	X	1	If format unsupported:ON	ELIB_SDFS_UNKNOWNFORMAT:0x01																																																																																									
X	X	X	X	X	X	1	X	If card size unsupported:ON	ELIB_SDFS_UNKNOWNCARDSIZE:0x02																																																																																									
X	X	X	X	X	1	X	X	If card type unsupported:ON	ELIB_SDFS_UNKNOWNCARDKIND:0x04																																																																																									
X	X	X	X	1	X	X	X	When write protected:ON	ELIB_SDFS_READY_WRITEPROTECT:0x08																																																																																									
X	X	X	1	X	X	X	X	While card recognition under process:ON	ELIB_SDFS_INITIAL_SDCARD:0x10																																																																																									
X	X	1	X	X	X	X	X	FAT entry: Error	ELIB_SDFS_UNUSUAL_FATENTRY:0x20																																																																																									
Include file		srv_sdfs.h																																																																																																
Calling sequence		int32_t Elib_SDFS_GetCardInfo(UINT Ap_ID, int8_t *cardinfo,uint8_t *CardID);																																																																																																
Argument		Type	I/O	Description																																																																																														
Ap_ID		UINT	I	Application ID																																																																																														
cardinfo		int8_t*	I/O	This argument specified card information.																																																																																														
CardID		uint8_t*	I/O	This argument specified card ID (16-byte array). The caller should allocate the field.																																																																																														
Return value		Type	-	Description																																																																																														
Ret		int32_t	O	- For normal end ELIB_SDFS_RETOK - For abnormal end (negative numeric) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing)																																																																																														
Remark																																																																																																		

10.28 SD use size acquisition function

Classification	SD management file service support function		
Function	Use size acquisition	Symbol	Elib_SDFS_GetUseSize
Function overview	<p>This function obtains the use size of an SD card.</p> <p>This function returns the number of bytes as the use size of an SD card.</p> <p>This function returns a value obtained by subtracting the total free area size of an SD card from the total size to a high-order process.</p>		
Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_GetUseSize(UINT Ap_ID);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
Return value	Type	-	Description
Ret	int32_t	O	<p>- For normal end 0to : Use size (unit: byte)</p> <p>- For abnormal end (negative numeric)</p> <p>ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error</p> <p>ELIB_SDFS_RETERR_IO: I-O error</p> <p>ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing)</p> <p>ELIB_SDFS_RETERR_MISC: Other errors</p>
Remark	<p>* Size "2G bytes" is not supported as SD card use size.</p>		

10.29 File detailed information acquisition

Classification	SD management file service support function		
Function	File detailed information acquisition	Symbol	Elib_SDFS_GetFileInfo
Function overview	This function obtains detailed file information.		
<div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><</div></div>			

(Termination character"\$0" contained)

- For the format of the save date structure, see the table below.

Save date structure format (ELIB_SDFS_DATE_DATA format)

Format and variable		Description
uint16_t	Year	Year (valid range depending on driver)
uint8_t	Mon	Month (1 to 12)
uint8_t	Day	Day (1 to 31)
uint8_t	Hour	Hour (0 to 23)
uint8_t	Min	Minute (0 to 59)
uint8_t	Sec	Second (0 to 59)
uint8_t	dumm	Dummy

Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_GetFileInfo(UINT Ap_ID, uint16_t DataID, uint16_t FileType, ELIB_SDFS_FILEINFO *list);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
DataID	uint16_t	I	This argument specifies a file.
FileType	uint16_t	I	This argument specifies a file type. ELIB_SDFS_VIDEO : Dynamic image ELIB_SDFS_PICTURE : Static image ELIB_SDFS_PIM :PIM
List	ELIB_SDFS_FILEINFO*	O	This argument specifies file detailed information structure (the caller should allocate the area.) * Details on the members are listed above.
Return value	Type	-	Description
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NODIR: File not found ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_BLOCKING: Blocking error ELIB_SDFS_RETERR_MISC: Other errors

Remark	

10.30 Parent folder information acquisition

Classification	SD management file service support function														
Function	Parent folder information acquisition	Symbol	Elib_SDFS_GetParentDirInfo												
Function overview	This function obtains information about a parent folder for a specified file/folder.														
<div><div>- Specify a value obtained with the Elib_SDFS_GetInfoList function in argument DataID.</div><div>- This function operates synchronously (in other words, it does not return unless its processing terminates).</div><div>- The caller should allocate a folder information (dirinfo) area.</div></div> <div>Valid ELIB_SDFS_DIRFILE_INFO members are listed below.</div> <div>No title for root folders obtained as parent folder information</div>															
<table><tr><th>Variable</th><th>Explanation</th></tr><tr><td>uint32_t Size</td><td>Folder/file size</td></tr><tr><td>uint16_t DataID</td><td>Data ID</td></tr><tr><td>uint16_t DirFileType</td><td>File type symbol <Folder type> ELIB_SDFS_DIR <File type> ELIB_SDFS_PICTURE_JPEG(static image, JPEG file) ELIB_SDFS_VIDEO_3GP(dynamic image, 3GP file) ELIB_SDFS_VIDEO_ASF(dynamic image, ASF file) ELIB_SDFS_VIDEO_SDV(dynamic image, SDV file) ELIB_SDFS_VIDEO_MP4(dynamic image, MP4 file) ELIB_SDFS_PIM_VCF(PIM, VCF file) ELIB_SDFS_PIM_VCS(PIM- VCS file) ELIB_SDFS_PIM_VMG(PIM- VMG file) ELIB_SDFS_PIM_VNT(PIM- VNT file) ELIB_SDFS_PIM_VBM(PIM- VBM file)</td></tr><tr><td>uint8_t DirFileName[64]</td><td>Name <Folder/file name mode>- - - 9 bytes Folder/file name (extension not contained in file name) + termination character (\$0) <title mode>- - - - - 64 bytes Folder/file title + termination character (\$0)</td></tr><tr><td>uint8_t Attr</td><td>Attribute Set the following data with or: ELIB_FA_NORMAL: Normal file ELIB_FA_READONLY: Read-only file ELIB_FA_HIDDEN: Hidden file ELIB_FA_SYSTEM: System file ELIB_FA_DIR: Subdirectory</td></tr></table>				Variable	Explanation	uint32_t Size	Folder/file size	uint16_t DataID	Data ID	uint16_t DirFileType	File type symbol <Folder type> ELIB_SDFS_DIR <File type> ELIB_SDFS_PICTURE_JPEG(static image, JPEG file) ELIB_SDFS_VIDEO_3GP(dynamic image, 3GP file) ELIB_SDFS_VIDEO_ASF(dynamic image, ASF file) ELIB_SDFS_VIDEO_SDV(dynamic image, SDV file) ELIB_SDFS_VIDEO_MP4(dynamic image, MP4 file) ELIB_SDFS_PIM_VCF(PIM, VCF file) ELIB_SDFS_PIM_VCS(PIM- VCS file) ELIB_SDFS_PIM_VMG(PIM- VMG file) ELIB_SDFS_PIM_VNT(PIM- VNT file) ELIB_SDFS_PIM_VBM(PIM- VBM file)	uint8_t DirFileName[64]	Name <Folder/file name mode>- - - 9 bytes Folder/file name (extension not contained in file name) + termination character (\$0) <title mode>- - - - - 64 bytes Folder/file title + termination character (\$0)	uint8_t Attr	Attribute Set the following data with or: ELIB_FA_NORMAL: Normal file ELIB_FA_READONLY: Read-only file ELIB_FA_HIDDEN: Hidden file ELIB_FA_SYSTEM: System file ELIB_FA_DIR: Subdirectory
Variable	Explanation														
uint32_t Size	Folder/file size														
uint16_t DataID	Data ID														
uint16_t DirFileType	File type symbol <Folder type> ELIB_SDFS_DIR <File type> ELIB_SDFS_PICTURE_JPEG(static image, JPEG file) ELIB_SDFS_VIDEO_3GP(dynamic image, 3GP file) ELIB_SDFS_VIDEO_ASF(dynamic image, ASF file) ELIB_SDFS_VIDEO_SDV(dynamic image, SDV file) ELIB_SDFS_VIDEO_MP4(dynamic image, MP4 file) ELIB_SDFS_PIM_VCF(PIM, VCF file) ELIB_SDFS_PIM_VCS(PIM- VCS file) ELIB_SDFS_PIM_VMG(PIM- VMG file) ELIB_SDFS_PIM_VNT(PIM- VNT file) ELIB_SDFS_PIM_VBM(PIM- VBM file)														
uint8_t DirFileName[64]	Name <Folder/file name mode>- - - 9 bytes Folder/file name (extension not contained in file name) + termination character (\$0) <title mode>- - - - - 64 bytes Folder/file title + termination character (\$0)														
uint8_t Attr	Attribute Set the following data with or: ELIB_FA_NORMAL: Normal file ELIB_FA_READONLY: Read-only file ELIB_FA_HIDDEN: Hidden file ELIB_FA_SYSTEM: System file ELIB_FA_DIR: Subdirectory														

uint8_t Multiflag		Whether folder/file with the same name is present ELIB_SDFS_MULTI_OFF: Not present ELIB_SDFS_MULTI_ON: Present	
uint8_t Dpofflag		DPOF flag status ELIB_SDFS_DPOF_OFF: Cleared ELIB_SDFS_DPOF_ON: Set	
uint8_t dummy		Dummy	
Include file		srv_sdfs.h	
Calling sequence		int32_t Elib_SDFS_GetParentDirInfo(UINT Ap_ID, uint16_t DataID ,uint16_t Namemode, ELIB_SDFS_DIRFILE_INFO *dirinfo);	
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
DataID	uint16_t	I	DataID
NameMode	uint16_t	I	Name mode <Folder/file name> ELIB_SDFS_DIRFILE_NAME <Title> ELIB_SDFS_TITLE_NAME
dirinfo	ELIB_SDFS_DIRFILE_INFO*	I/O	Folder information area parameter (The caller should allocate the area.)
Return value	Type	-	Description
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK - For normal end ELIB_SDFS_RETOK - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NODIR: Folder not found ELIB_SDFS_RETERR_NODIR: File not found ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_BLOCKING: Blocking error ELIB_SDFS_RETERR_MISC: Other errors
Remark			

10.31 File presence check

Classification	SD management file service support function																				
Function	File presence check	Symbol	Elib_SDFS_CheckDir																		
Function overview	This function checks if a file is present in a specified folder.																				
<div>- This function operates synchronously (in other words, it does not return unless its processing terminates).</div> <div>- The caller should allocate a Filechkflg field.</div> <div>- A specified folder need be initialized (Elib_SDFS_DirInitAsync).</div> <div>- Check results set in an argument (Filechkflg) are listed below.</div>																					
<table><tr><th>Symbol</th><th>Explanation</th><th>Value</th></tr><tr><td>ELIB_SDFS_EmptyDir</td><td>No file is found.</td><td>00000000 bit</td></tr><tr><td>ELIB_SDFS_DelFile</td><td>A file that can be deleted is found.</td><td>00000001 bit</td></tr><tr><td>ELIB_SDFS_NonDelFile</td><td>A file that cannot be deleted is found.</td><td>00000010 bit</td></tr><tr><td>ELIB_SDFS_ReadOnly</td><td>A read-only file is found.</td><td>00000100 bit</td></tr><tr><td>ELIB_SDFS_ReadOnlyALL</td><td>Only a read-only file is found.</td><td>10000000 bit</td></tr></table>				Symbol	Explanation	Value	ELIB_SDFS_EmptyDir	No file is found.	00000000 bit	ELIB_SDFS_DelFile	A file that can be deleted is found.	00000001 bit	ELIB_SDFS_NonDelFile	A file that cannot be deleted is found.	00000010 bit	ELIB_SDFS_ReadOnly	A read-only file is found.	00000100 bit	ELIB_SDFS_ReadOnlyALL	Only a read-only file is found.	10000000 bit
Symbol	Explanation	Value																			
ELIB_SDFS_EmptyDir	No file is found.	00000000 bit																			
ELIB_SDFS_DelFile	A file that can be deleted is found.	00000001 bit																			
ELIB_SDFS_NonDelFile	A file that cannot be deleted is found.	00000010 bit																			
ELIB_SDFS_ReadOnly	A read-only file is found.	00000100 bit																			
ELIB_SDFS_ReadOnlyALL	Only a read-only file is found.	10000000 bit																			
Include file	srv_sdfs.h																				
Calling sequence	int32_t Elib_SDFS_CheckDir(UINT Ap_ID, uint16_t DataID, uint16_t DirType, uint8_t *Filechkflg);																				
Argument	Type	I/O	Description																		
Ap_ID	UINT	I	Application ID																		
DataID	uint16_t	I	This argument specified folder ID as follows: <Static image/dynamic image> Specify data ID obtained with the folder/file list information acquisition function (No.25/35). <PIM> (Telephone directory) :ELIB_SDFS_ROOTDIR_CARD (Schedule) :ELIB_SDFS_ROOTDIR_CALEDAR (Received mail) :ELIB_SDFS_ROOTDIR_INBOX (Untransmitted mail) :ELIB_SDFS_ROOTDIR_OUTBOX (Transmitted mail) :ELIB_SDFS_ROOTDIR_SENTBOX (Free memorandum) :ELIB_SDFS_ROOTDIR_NOTE (Bookmark) :ELIB_SDFS_ROOTDIR_BOOKMARK																		
DirType	uint16_t	I	This argument specified a folder type as follows: <Static image> ELIB_SDFS_PICTURE																		

			<u><Dynamic image></u> ELIB_SDFS_VIDEO <u><PIM></u> (Telephone directory) :ELIB_SDFS_PIM_CARD (Schedule) :ELIB_SDFS_PIM_CALENDAR (Received mail) :ELIB_SDFS_PIM_INBOX (Untransmitted mail) :ELIB_SDFS_PIM_OUTBOX (Transmitted mail) :ELIB_SDFS_PIM_SENTBOX (Free memorandum) :ELIB_SDFS_PIM_NOTE (Bookmark) :ELIB_SDFS_PIM_BOOKMARK
Filechkflg	uint8_t*	I/O	
Return value	Type	-	Description
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NOTINIT: Folder uninitialized ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_BLOCKING: Blocking error ELIB_SDFS_RETERR_MISC: Other errors
Remark			

10.32 Information for indicator acquiring function

Classification	SD management file service support function		
Function	Information for indicator acquiring	Symbol	Elib_SDFS_GetIndicatorInfo
Function overview	<p>This function obtains processing progress information.</p> <ul style="list-style-type: none"> - This function operates synchronously (in other words, it does not return unless its processing terminates). - Call this function during processing. - The caller can grasp processing progress by checking the number of processed items/the total number of items to be processed. 		
Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_GetIndicatorInfo(UINT Ap_ID, uint16_t *endnum, uint16_t *totalnum);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
endnum	uint16_t*	O	This argument indicates the number of processed items.
totalnum	uint16_t*	O	This argument indicates the total number of items to be processed.
Return value	Type	-	Description
Ret	int32_t	O	<ul style="list-style-type: none"> - For normal end ELIB_SDFS_RETOK - For abnormal end (negative numeric) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors
Remark			

10.33 Card size information acquisition

Classification	SD management file service support function		
Function	Card size information acquisition	Symbol	Elib_SDFS_GetCardSizeInfo
Function overview	<p>This function returns the total size and total free area size of an SD card by using the number of bytes.</p> <p>- This function returns the total size and total free area size of an SD card.</p>		
Include file	srv_sdfs.h		
Calling sequence	<pre>int32_t Elib_SDFS_GetCardSizeInfo(UINT Ap_ID, uint32_t *totalsize, uint32_t *freesize);</pre>		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
Totalsize	uint32_t*		<p>This argument indicates the total size of an SD card (number of bytes).</p> <p>(The caller should allocate the storage field.)</p>
Freesize	uint32_t*		<p>This argument indicates the total free area size of an SD card (number of bytes).</p> <p>(The caller should allocate the storage field.)</p>
Return value	Type	-	Description
Ret	int32_t	O	<p>- For normal end ELIB_SDFS_RETOK</p> <p>- For abnormal end (negative numeric) ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_MISC: Other errors</p>
Remarks	<p>* The total size, total free area size, and total busy area size of an SD card can be obtained by using this function.</p> <p>* Use of this function instead of the Elib_SDFS_GetUseSize and Elib_SDFS_GetFreeSize functions is recommended in future operation with size 2G supported.</p>		

10.34 Save destination folder data ID acquisition

Classification	SD management file service support function		
Function	Save destination folder data ID acquisition	Symbol	Elib_SDFS_GetSaveDirID
Function overview	<p>This function obtains the data ID of a save destination folder.</p> <p>- This function obtains the data ID of a save destination folder.</p>		
Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_GetSaveDirID (UINT Ap_ID, uint16_t FileType, uint16_t *SaveDir);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
FileType	uint16_t	I	This argument specifies a file type as follows: ELIB_SDFS_VIDEO : Dynamic image ELIB_SDFS_PICTURE : Static image
SaveDir	uint16_t*	I/O	This argument specifies a save destination folder. The caller should allocate the field.
Return value	Type	-	Description
Ret	int32_t	O	<p>- For normal end ELIB_SDFS_RETOK</p> <p>- For abnormal end (negative numeric) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_MISC: Other errors</p>
Remarks			

10.35 SD card insertion status acquisition

Classification	SD management file service support function		
Function	SD card insertion status acquisition	Symbol	Elib_SDFS_GetCardInInfo
Function overview	This function obtains SD card insertion status information.		
<p>This function returns a value indicating one of the following SD card insertion statuses:</p> <ul style="list-style-type: none">- No SD card is inserted.- An SD card is inserted.- An SD card is inserted (with write protected).- An SD card is inserted (error). <p>If the above information cannot be obtained, this function returns a value indicating abnormal end (negative numeric).</p>			
Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_GetCardInInfo(UINT Ap_ID);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
Return value	Type	-	Description
Ret	int32_t	O	<p>- For normal end (positive numeric)</p> <p>ELIB_SDFS_CARD_NOEXIST: SD card not inserted</p> <p>ELIB_SDFS_CARD_EXIST: SD card inserted</p> <p>ELIB_SDFS_CARD_EXIST_WP: SD card inserted (write protected)</p> <p>ELIB_SDFS_CARD_EXIST_ERR: SD card inserted (error)</p> <p>- For abnormal end (negative numeric)</p> <p>ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error</p> <p>ELIB_SDFS_RETERR_MISC: Other errors</p> <p>ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing)</p>
Remark			

10.36 File detailed information acquisition (asynchronous)

Classification	SD management file service support function														
Function	File detailed information acquisition	Symbol	Elib_SDFS_GetFileInfoAsync												
Function overview	This function obtains detailed information about a specified file (DataID) (asynchronously).														
<div>- This function is an asynchronous version of the Elib_SDFS_GetFileInfo function (10.29).</div> <div>- For obtained information, reference an area allocated at the calling of this API when receiving an event indicating normal end.</div> <div>- No event occurs unless this function terminates normally (processing starts) (return value: ELIB_SDFS_RETOK).</div>															
After terminating information acquisition, this function creates and sends events listed below (info, subinfo = additional information in event transmission data).															
<table><tr><th>Event symbol</th><th>info (transmission data)</th><th>subinfo (transmission data)</th><th>Explanation</th></tr><tr><td>SDFSNotify_INFOGET_END</td><td>SDFSNotify_INFOGET_END</td><td>ELIB_SDFS_RETOK</td><td>Normal end</td></tr><tr><td>SDFSNotify_INFOGET_END</td><td>SDFSNotify_INFOGET_END</td><td>Error type (negative)</td><td>Abnormal end</td></tr></table>				Event symbol	info (transmission data)	subinfo (transmission data)	Explanation	SDFSNotify_INFOGET_END	SDFSNotify_INFOGET_END	ELIB_SDFS_RETOK	Normal end	SDFSNotify_INFOGET_END	SDFSNotify_INFOGET_END	Error type (negative)	Abnormal end
Event symbol	info (transmission data)	subinfo (transmission data)	Explanation												
SDFSNotify_INFOGET_END	SDFSNotify_INFOGET_END	ELIB_SDFS_RETOK	Normal end												
SDFSNotify_INFOGET_END	SDFSNotify_INFOGET_END	Error type (negative)	Abnormal end												
<div>- Error types are listed below.</div> <table><tr><th>Symbol</th><th>Type</th></tr><tr><td>ELIB_SDFS_RETERR_NOFILE</td><td>No file</td></tr><tr><td>ELIB_SDFS_RETERR_ILLEGAL_PARAM</td><td>Parameter error</td></tr><tr><td>ELIB_SDFS_RETERR_IO</td><td>I-O error</td></tr><tr><td>ELIB_SDFS_RETERR_ABORTCALLED</td><td>Halt with halt request</td></tr><tr><td>ELIB_SDFS_RETERR_MISC</td><td>Others</td></tr></table>				Symbol	Type	ELIB_SDFS_RETERR_NOFILE	No file	ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error	ELIB_SDFS_RETERR_IO	I-O error	ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request	ELIB_SDFS_RETERR_MISC	Others
Symbol	Type														
ELIB_SDFS_RETERR_NOFILE	No file														
ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error														
ELIB_SDFS_RETERR_IO	I-O error														
ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request														
ELIB_SDFS_RETERR_MISC	Others														
The data below is passed as structure of data sent via the event transmitter (see Section 4.4).															
int shmid;															
int size;															
Access a shared memory using the above parameters to obtain file detailed information structure data.															
The calling API side should release the shared memory after obtaining data.															
Type of file detailed information structure data:															
ELIB_SDFS_FILEINFO															
File detailed information structure: For the members, see 10.29, “File detailed information acquisition”.															
Include file	srv_sdfs.h														
Calling sequence	int32_t Elib_SDFS_GetFileInfoAsync(UINT Ap_ID, uint16_t DataID, uint16_t FileType);														

Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
DataID	uint16_t	I	This argument specifies a file data ID.
FileType	uint16_t	I	This argument specifies a file type as follows: ELIB_SDFS_VIDEO : Dynamic image ELIB_SDFS_PICTURE : Static image ELIB_SDFS_PIM :PIM
Return value	Type	-	Description
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK - For abnormal end (negative numeric) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_NOFILE: File not found ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing)
Remark	* See 10.24, "Total folder/file list count acquisition".		

10.37 Folder/file list information acquisition (asynchronous)

Classification	SD management file service support function																
Function	Folder/file information acquisition	Symbol	Elib_SDFS_GetInfoListAsync														
Function overview	This function obtains a detailed information list of files under a specified folder and folders lower hierarchically than it.																
<div>- This function is an asynchronous version of the Elib_SDFS_GetInfoList function (10.25).</div> <div>- For obtained information, reference an area allocated at the calling of this API when receiving an event indicating normal end.</div> <div>- The specified folder need be initialized (Elib_SDFS_DirInitAsync).</div> <div>- No event occurs unless this function terminates normally (processing starts) (return value: ELIB_SDFS_RETOK).</div>																	
After terminating information acquisition, this function creates and sends events listed below (info, subinfo = additional information in event transmission data).																	
<table><tr><th>Event symbol</th><th>info (transmission data)</th><th>subinfo (transmission data)</th><th>Explanation</th></tr><tr><td>SDFSNotify_LIST_INFOGET_END</td><td>SDFSNotify_LIST_INFOGET_END</td><td>ELIB_SDFS_RETOK</td><td>Normal end</td></tr><tr><td>SDFSNotify_LIST_INFOGET_END</td><td>SDFSNotify_LIST_INFOGET_END</td><td>Error type (negative)</td><td>Abnormal end</td></tr></table>				Event symbol	info (transmission data)	subinfo (transmission data)	Explanation	SDFSNotify_LIST_INFOGET_END	SDFSNotify_LIST_INFOGET_END	ELIB_SDFS_RETOK	Normal end	SDFSNotify_LIST_INFOGET_END	SDFSNotify_LIST_INFOGET_END	Error type (negative)	Abnormal end		
Event symbol	info (transmission data)	subinfo (transmission data)	Explanation														
SDFSNotify_LIST_INFOGET_END	SDFSNotify_LIST_INFOGET_END	ELIB_SDFS_RETOK	Normal end														
SDFSNotify_LIST_INFOGET_END	SDFSNotify_LIST_INFOGET_END	Error type (negative)	Abnormal end														
<div>- Error types are listed below.</div> <table><tr><th>Symbol</th><th>Type</th></tr><tr><td>ELIB_SDFS_RETERR_NODIR</td><td>No folder</td></tr><tr><td>ELIB_SDFS_RETERR_NOFILE</td><td>No file</td></tr><tr><td>ELIB_SDFS_RETERR_ILLEGAL_PARAM</td><td>Parameter error</td></tr><tr><td>ELIB_SDFS_RETERR_IO</td><td>I-O error</td></tr><tr><td>ELIB_SDFS_RETERR_ABORTCALLED</td><td>Halt with halt request</td></tr><tr><td>ELIB_SDFS_RETERR_MISC</td><td>Others</td></tr></table>				Symbol	Type	ELIB_SDFS_RETERR_NODIR	No folder	ELIB_SDFS_RETERR_NOFILE	No file	ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error	ELIB_SDFS_RETERR_IO	I-O error	ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request	ELIB_SDFS_RETERR_MISC	Others
Symbol	Type																
ELIB_SDFS_RETERR_NODIR	No folder																
ELIB_SDFS_RETERR_NOFILE	No file																
ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error																
ELIB_SDFS_RETERR_IO	I-O error																
ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request																
ELIB_SDFS_RETERR_MISC	Others																
The data below is passed as structure of data sent via the event transmitter (see Section 4.4).																	
<div>int shmid;</div> <div>int size;</div> <div>Access a shared memory using the above parameters to obtain file/folder detailed information structure data.</div> <div>The calling API side should release the shared memory after obtaining data.</div>																	
Type of file/folder detailed information structure data: ELIB_SDFS_DIRFILE_INFO																	
Include file	srv_sdfs.h																
Calling sequence	int32_t Elib_SDFS_GetInfoListAsync(UINT Ap_ID, uint16_t DataID, uint16_t DirFileType, uint16_t Namemode, uint16_t Direct, uint16_t GetStartNumber,																

			uint16_t GetNumber);
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
DataID	uint16_t	I	<p>This argument specifies a folder data ID as follows:</p> <p>Folder under a directory in hierarchy not higher than that of DCIM directory:ELIB_SDFS_ROOTDIR_DCIM</p> <p>Folder under a directory in hierarchy not higher than that of SD-VIDEO directory:ELIB_SDFS_ROOTDIR_SDVIDEO</p> <p>Folder under a directory in hierarchy not higher than SD-PIM directory:ELIB_SDFS_ROOTDIR_SDPIM</p>
DirFileType	uint16_t	I	<p>This argument specifies a folder as follows:.</p> <p>ELIB_SDFS_DIR</p> <p>File type symbol</p> <p>ELIB_SDFS_PICTURE General static mages stored in SD card</p> <p>ELIB_SDFS_PICTURE_JPEG JPEG stored in SD card</p> <p>ELIB_SDFS_VIDEO General dynamic images stored in SD card</p> <p>ELIB_SDFS_VIDEO_ASF ASF dynamic image stored in SD card</p> <p>ELIB_SDFS_VIDEO_3GP 3GP dynamic image stored in SD card</p> <p>ELIB_SDFS_VIDEO_SDV SDV dynamic image stored in SD card</p> <p>ELIB_SDFS_VIDEO_MP4 MP4 dynamic image stored in SD card</p> <p>ELIB_SDFS_PIM : PIM stored in SD card</p> <p>ELIB_SDFS_PIM_CARD:vCard</p> <p>ELIB_SDFS_PIM_CALENDAR:vCalendar</p> <p>ELIB_SDFS_PIM_INBOX: Received mail</p> <p>ELIB_SDFS_PIM_OUTBOX: Untransmitted mail</p> <p>ELIB_SDFS_PIM_SENTBOX: Transmitted mail</p> <p>ELIB_SDFS_PIM_NOTE: Free memorandum</p> <p>ELIB_SDFS_PIM_BOOKMARK: Bookmark</p>
NameMode	uint16_t	I	<p>This argument specifies name mode.</p> <p>Name for list display</p> <p><Folder/file name> ELIB_SDFS_DIRFILE_NAME</p> <p><Title> ELIB_SDFS_TITLE_NAME</p>
Direct	uint16_t	I	<p>This argument specifies an acquisition direction as follows:</p> <p>0: Smallest No. to Largest No.</p> <p>1: Largest No. to smallest No.</p> <p>For PIM, acquired order is specified automatically.</p>

GetStartNumber	uint16_t	I	This argument specifies an acquisition start number: 1 (data displayed at the top) to n
GetNumber	uint16_t	I	This argument specifies the number of lists to be obtained.
Return value	Type	-	Description
Ret	int32_t	O	<ul style="list-style-type: none"> - For normal end ELIB_SDFS_RETOK: Normal end (processing start) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_NODIR: Folder not found ELIB_SDFS_RETERR_NOTINI: Folder not initialized ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing)
Remark	<p>* See 10.25, "Folder/file list information acquisition".</p>		

10.38 vObject list information acquisition

Classification	SD management file service support function		
Function	vObject list information acquisition	Symbol	Elib_SDFS_GetvObjectInfoListAsync
Function overview	This function obtains vObject list information in a specified PIM file (asynchronously).		

- No event occurs unless this function terminates normally (processing starts) (return value: ELIB_SDFS_RETOK).
- The caller should allocate an area for t_vobjinfo.
- For obtained information, reference an area allocated at the calling of this API when receiving an event indicating normal end.
- The T_SRV_SDPIME_REQ_VOBJINFO format is shown below.

Variable	I/O	Explanation
unsigned char vobjinfo	I	Type of target vObject - ELIB_SDFS_CARD vCard - ELIB_SDFS_CALENDAR vCalendar - ELIB_SDFS_INBOX Received mail - ELIB_SDFS_OUTBOX Untransmitted mail - ELIB_SDFS_SENTBOX Transmitted mail - ELIB_SDFS_NOTE Free memorandum - ELIB_SDFS_BOOKMARK Bookmark
unsigned char reserved	-	Boundary
unsigned short id	I	ID of target file
unsigned short reqcnt	I	Number of items obtained at once
unsigned short titlesi	I	Target title size
signed long offset	I	Read position (offset from the top of target file)
unsigned char getvobj	O	Unused
signed long pos	O	Unused
signed long vobjsi	O	Unused
unsigned char title	O	Unused

After terminating acquisition, this function creates and sends events listed below (info, subinfo = additional information in event transmission data).

Event symbol	info (transmission data)	subinfo (transmission data)	Explanation
SDFSNotify_vObject_LIST_IN NFOGET_END	SDFSNotify_vObject_LIST_IN FOGET_END	ELIB_SDFS_RE TOK	Normal end
SDFSNotify_vObject_LIST_IN NFOGET_END	SDFSNotify_vObject_LIST_IN FOGET_END	Error type (negative)	Abnormal end

- Error types are listed below.

Symbol	Type
ELIB_SDFS_RETERR_NODIR	No directory
ELIB_SDFS_RETERR_NOFILE	No file
ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error
ELIB_SDFS_RETERR_IO	I-O error
ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request
ELIB_SDFS_RETERR_MISC	Others

The data below is passed as structure of data sent via the event transmitter (see Section 4.4).

```
int shmid;
uint32_t size;
```

```
uint16_t no; /* number of obtained items */
```

Access a shared memory using the above parameters to obtain data on obtained information.

The calling API side should release the shared memory after obtaining the data.

In the obtained data, structures (the format is shown below) as many as the number of obtained information items are arranged.

```
typedef struct eilib_sdfs_vobjinfo_output {
    signed long    pos; /* vObject off set from file top */
    signed long    vobjsize; /* size of obtained vObject */
    unsigned char  title[64]; /*storage field of obtained title */
    unsigned char  getvobj; /* Type of obtained vObject */
    unsigned char  dmy[3]; /* Unused */
} ELIB_SDFS_VOBJINFO_OUTPUT
```

Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_GetvObjectInfoListAsync(UINT Ap_ID, T_SRV_SDPIPE_REQ_VOBJINFO *t_vobjinfo);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
t_vobjinfo	T_SRV_SDPIPE_REQ_VO	I	This argument indicates the pointer to

	BJINFO*		vObject list information structure (The caller should allocate the field.)
Return value	Type	-	Description
Ret	int32_t	O	<ul style="list-style-type: none"> - For normal end ELIB_SDFS_RETOK: Normal end (processing start) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing)
Remark			

10.39 Detailed vObject information acquisition

Classification	SD management file service support function																														
Function	Detailed vObject information acquisition	Symbol	Elib_SDFS_GetvObjectDataAsync																												
Function overview	This function obtains detailed information about specified vObject (asynchronously).																														
<div>- The caller should allocate an area for t_vobjget.</div> <div>- For obtained information, reference an area allocated at the calling of this API when receiving an event indicating normal end.</div> <div>- The T_SRV_SDPIME_REQ_VOBJGET format is shown below.</div>																															
<table><tr><th>Variable</th><th>I/O</th><th colspan="2">Explanation</th></tr><tr><td>unsigned char vobjinfo</td><td>I</td><td colspan="2">Type of target vObject - ELIB_SDFS_CARD vCard - ELIB_SDFS_EVENT vEvent - ELIB_SDFS_TODO vTODO - ELIB_SDFS_EMAIL_IN Received e-mail - ELIB_SDFS_EMAIL_OUT Untransmitted e-mail - ELIB_SDFS_EMAIL_SENT Transmitted e-mail - ELIB_SDFS_SMS_IN Received SMS-mail - ELIB_SDFS_SMS_OUT Untransmitted SMS-mail - ELIB_SDFS_SMS_SENT Transmitted SMS-mail - ELIB_SDFS_NOTE Free memorandum - ELIB_SDFS_BOOKMARK Bookmark</td></tr><tr><td>unsigned char reserved</td><td>-</td><td colspan="2">Boundary</td></tr><tr><td>unsigned short id</td><td>I</td><td colspan="2">ID of target file</td></tr><tr><td>signed long pos</td><td>I</td><td colspan="2">Target vObject position (obtained with the vObject list information acquisition function (Elib_SDFS_GetvObjectInfoListAsync))</td></tr><tr><td>signed long vobjsize</td><td>I</td><td colspan="2">Size of target vObject (vobjsize obtained with the vObject list information acquisition function (Elib_SDFS_GetvObjectInfoListAsync))</td></tr><tr><td>void* data</td><td>O</td><td colspan="2">Unused</td></tr></table>				Variable	I/O	Explanation		unsigned char vobjinfo	I	Type of target vObject - ELIB_SDFS_CARD vCard - ELIB_SDFS_EVENT vEvent - ELIB_SDFS_TODO vTODO - ELIB_SDFS_EMAIL_IN Received e-mail - ELIB_SDFS_EMAIL_OUT Untransmitted e-mail - ELIB_SDFS_EMAIL_SENT Transmitted e-mail - ELIB_SDFS_SMS_IN Received SMS-mail - ELIB_SDFS_SMS_OUT Untransmitted SMS-mail - ELIB_SDFS_SMS_SENT Transmitted SMS-mail - ELIB_SDFS_NOTE Free memorandum - ELIB_SDFS_BOOKMARK Bookmark		unsigned char reserved	-	Boundary		unsigned short id	I	ID of target file		signed long pos	I	Target vObject position (obtained with the vObject list information acquisition function (Elib_SDFS_GetvObjectInfoListAsync))		signed long vobjsize	I	Size of target vObject (vobjsize obtained with the vObject list information acquisition function (Elib_SDFS_GetvObjectInfoListAsync))		void* data	O	Unused	
Variable	I/O	Explanation																													
unsigned char vobjinfo	I	Type of target vObject - ELIB_SDFS_CARD vCard - ELIB_SDFS_EVENT vEvent - ELIB_SDFS_TODO vTODO - ELIB_SDFS_EMAIL_IN Received e-mail - ELIB_SDFS_EMAIL_OUT Untransmitted e-mail - ELIB_SDFS_EMAIL_SENT Transmitted e-mail - ELIB_SDFS_SMS_IN Received SMS-mail - ELIB_SDFS_SMS_OUT Untransmitted SMS-mail - ELIB_SDFS_SMS_SENT Transmitted SMS-mail - ELIB_SDFS_NOTE Free memorandum - ELIB_SDFS_BOOKMARK Bookmark																													
unsigned char reserved	-	Boundary																													
unsigned short id	I	ID of target file																													
signed long pos	I	Target vObject position (obtained with the vObject list information acquisition function (Elib_SDFS_GetvObjectInfoListAsync))																													
signed long vobjsize	I	Size of target vObject (vobjsize obtained with the vObject list information acquisition function (Elib_SDFS_GetvObjectInfoListAsync))																													
void* data	O	Unused																													
<div>- No event occurs unless this function terminates normally (processing starts) (return value: ELIB_SDFS_RETOK).</div> <div>After terminating import, this function creates and sends events listed below (info, subinfo = additional information in event transmission data).</div> <table><tr><th>Event symbol</th><th>info (transmission data)</th><th>subinfo (transmission data)</th><th>Explanation</th></tr><tr><td>SDFSNotify_vObject_INFOGET_END</td><td>SDFSNotify_vObject_INFOGET_END</td><td>ELIB_SDFS_RETOK</td><td>Normal end</td></tr><tr><td>SDFSNotify_vObject_INFOGET_END</td><td>SDFSNotify_vObject_INFOGET_END</td><td>Error type (negative)</td><td>Abnormal end</td></tr></table> <div>- Error types are listed below.</div> <table><tr><th>Symbol</th><th>Type</th></tr></table>				Event symbol	info (transmission data)	subinfo (transmission data)	Explanation	SDFSNotify_vObject_INFOGET_END	SDFSNotify_vObject_INFOGET_END	ELIB_SDFS_RETOK	Normal end	SDFSNotify_vObject_INFOGET_END	SDFSNotify_vObject_INFOGET_END	Error type (negative)	Abnormal end	Symbol	Type														
Event symbol	info (transmission data)	subinfo (transmission data)	Explanation																												
SDFSNotify_vObject_INFOGET_END	SDFSNotify_vObject_INFOGET_END	ELIB_SDFS_RETOK	Normal end																												
SDFSNotify_vObject_INFOGET_END	SDFSNotify_vObject_INFOGET_END	Error type (negative)	Abnormal end																												
Symbol	Type																														

ELIB_SDFS_RETERR_NODIR	No folder
ELIB_SDFS_RETERR_NOFILE	No file
ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error
ELIB_SDFS_RETERR_IO	I-O error
ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request
ELIB_SDFS_RETERR_MISC	Others

The data below is passed as structure of data sent via the event transmitter (see Section 4.4).

int shmid;

int size;

Access a shared memory using the above parameters to obtain data in the mobile phone.

The calling API side should release the shared memory after obtaining the data.

Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_GetvObjectDataAsync(UINT Ap_ID, T_SRV_SDPIME_REQ_VOBJGET *t_vobjget);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
t_vobjget	T_SRV_SDPIME_REQ_VOBJGET *	I	This argument indicates the pointer to vObject detailed information structure (The caller should allocate the field.)
Return value	Type	-	Description
Ret	int32_t	O	<ul style="list-style-type: none"> - For normal end ELIB_SDFS_RETOK: Normal end (processing start) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing)
Remark			

10.40 Check disk

Classification	SD management file service support function		
Function	Check disk	Symbol	Elib_SDFS_ChkDsk
Function overview	<p>This function makes a disk check.</p> <ul style="list-style-type: none"> - A disk check is made with an API function. - If an error is detected, it can be corrected depending on the Repair parameter. The management table is also corrected in error correction. - If an error is detected, event ELIB_SDFS_RETERR_CHKDSK is generated. - This function operates synchronously (in other words, it does not return unless its processing terminates). 		
Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_ChkDsk(UINT Ap_ID, int32_t Repair);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
Ripair	int32_t	I	ELIB_SDFS_CHKDSK_ONLY (only check (without error correction)) ELIB_SDFS_CHKDSK_MODIFY (Detected error is corrected.)
Return value	Type	-	Description
Ret	int32_t	O	<ul style="list-style-type: none"> - For normal end Normal end:ELIB_SDFS_RETOK - For abnormal end (negative numeric) ELIB_SDFS_RETERR_CHKDSK: Check disk error ELIB_SDFS_RETERR_WRITEPROTECT: Write protect error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_MISC: Other error ELIB_SDFS_RETERR_ABORTCALLED: Halt with halt request
Remark			

10.41 SD card format (asynchronous)

Classification	SD management file service support function																								
Function	SD format card	Symbol	Elib_SDFS_FormatCardAsync																						
Function overview	This function formats an SD card (asynchronously).																								
<div>- This function operates asynchronously.</div> <div>ELIB_SDFS_QUICKFORMAT: Quick format</div> <div>ELIB_SDFS_NORMALFORMAT: Normal format</div> <div>- No event occurs unless this function terminates normally (processing starts) (return value: ELIB_SDFS_RETOK).</div> <div>After terminating format, this function creates and sends events listed below (info, subinfo = additional information in event transmission data).</div> <table><tr><th>Event symbol</th><th>info (transmission data)</th><th>subinfo (transmission data)</th><th>Explanation</th></tr><tr><td>SDFSNotify_FORMAT_EN D</td><td>SDFSNotify_FORMAT_EN D</td><td>ELIB_SDFS_RETOK</td><td>Normal end</td></tr><tr><td>SDFSNotify_FORMAT_EN D</td><td>SDFSNotify_FORMAT_EN D</td><td>Error type (negative)</td><td>Abnormal end</td></tr></table> <div>- Error types are listed below.</div> <table><tr><th>Symbol</th><th>Type</th></tr><tr><td>ELIB_SDFS_RETERR_ILLEGAL_PARAM</td><td>Parameter error</td></tr><tr><td>ELIB_SDFS_RETERR_IO</td><td>I-O error</td></tr><tr><td>ELIB_SDFS_RETERR_ABORTCALLED</td><td>Halt with halt request</td></tr><tr><td>ELIB_SDFS_RETERR_MISC</td><td>Others</td></tr></table>				Event symbol	info (transmission data)	subinfo (transmission data)	Explanation	SDFSNotify_FORMAT_EN D	SDFSNotify_FORMAT_EN D	ELIB_SDFS_RETOK	Normal end	SDFSNotify_FORMAT_EN D	SDFSNotify_FORMAT_EN D	Error type (negative)	Abnormal end	Symbol	Type	ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error	ELIB_SDFS_RETERR_IO	I-O error	ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request	ELIB_SDFS_RETERR_MISC	Others
Event symbol	info (transmission data)	subinfo (transmission data)	Explanation																						
SDFSNotify_FORMAT_EN D	SDFSNotify_FORMAT_EN D	ELIB_SDFS_RETOK	Normal end																						
SDFSNotify_FORMAT_EN D	SDFSNotify_FORMAT_EN D	Error type (negative)	Abnormal end																						
Symbol	Type																								
ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error																								
ELIB_SDFS_RETERR_IO	I-O error																								
ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request																								
ELIB_SDFS_RETERR_MISC	Others																								
Include file	srv_sdfs.h																								
Calling sequence	int32_t Elib_SDFS_FormatCardAsync(UINT Ap_ID, int32_t Option, const uint8_t *VolumeName);																								
Argument	Type	I/O	Description																						
Ap_ID	UINT	I	Application ID																						
Option	int32_t	I	ELIB_SDFS_QUICKFORMAT: Quick format ELIB_SDFS_NORMALFORMAT: Normal format																						
VolumeName	const uint8_t*	I	This argument specifies the volume name of a target card. The default is NULL.																						
Return value	Type	-	Description																						
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK: Normal end (processing start) - For abnormal end (negative numeric)																						

		<div>ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error</div> <div>ELIB_SDFS_RETERR_MISC: Error other than parameter error</div> <div>ELIB_SDFS_RETERR_IO : I-O error</div> <div>ELIB_SDFS_RETERR_BLOCKING: Blocking error</div> <div>ELIB_SDFS_RETERR_WRITEPROTECT : Write protect error</div> <div>ELIB_SDFS_RETERR_ABORTCALLED: Halt with halt request</div>
Remark		

10.42 Check disk (asynchronous)

Classification	SD management file service support function														
Function	Check disk	Symbol	Elib_SDFS_ChkDskAsync												
Function overview	This function is an asynchronous version of the Elib_SDFS_ChkDsk function (10.38).														
<div>- This function makes a disk check.</div> <div>- If an error is detected, it can be corrected depending on the Repair parameter. The management table is also corrected in error correction.</div> <div>- If an error is detected, event ELIB_SDFS_RETERR_CHKDSK is generated.</div> <div>- This function operates asynchronously.</div> <div>- No event occurs unless this function terminates normally (processing starts) (return value: ELIB_SDFS_RETOK).</div>															
After terminating check, this function creates and sends events listed below (info, subinfo = additional information in event transmission data).															
<table><tr><th>Event symbol</th><th>info (transmission data)</th><th>subinfo (transmission data)</th><th>Explanation</th></tr><tr><td>SDFSNotify_CHKDSK_END</td><td>SDFSNotify_CHKDSK_END</td><td>ELIB_SDFS_RETOK</td><td>Normal end</td></tr><tr><td>SDFSNotify_CHKDSK_END</td><td>SDFSNotify_CHKDSK_END</td><td>Error type (negative)</td><td>Abnormal end</td></tr></table>				Event symbol	info (transmission data)	subinfo (transmission data)	Explanation	SDFSNotify_CHKDSK_END	SDFSNotify_CHKDSK_END	ELIB_SDFS_RETOK	Normal end	SDFSNotify_CHKDSK_END	SDFSNotify_CHKDSK_END	Error type (negative)	Abnormal end
Event symbol	info (transmission data)	subinfo (transmission data)	Explanation												
SDFSNotify_CHKDSK_END	SDFSNotify_CHKDSK_END	ELIB_SDFS_RETOK	Normal end												
SDFSNotify_CHKDSK_END	SDFSNotify_CHKDSK_END	Error type (negative)	Abnormal end												
<div>- Error types are listed below.</div> <table><tr><th>Symbol</th><th>Type</th></tr><tr><td>ELIB_SDFS_RETERR_CHKDSK</td><td>Check disk error</td></tr><tr><td>ELIB_SDFS_RETERR_ILLEGAL_PARAM</td><td>Parameter error</td></tr><tr><td>ELIB_SDFS_RETERR_IO</td><td>I-O error</td></tr><tr><td>ELIB_SDFS_RETERR_ABORTCALLED</td><td>Halt with halt request</td></tr><tr><td>ELIB_SDFS_RETERR_MISC</td><td>Others</td></tr></table>				Symbol	Type	ELIB_SDFS_RETERR_CHKDSK	Check disk error	ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error	ELIB_SDFS_RETERR_IO	I-O error	ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request	ELIB_SDFS_RETERR_MISC	Others
Symbol	Type														
ELIB_SDFS_RETERR_CHKDSK	Check disk error														
ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error														
ELIB_SDFS_RETERR_IO	I-O error														
ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request														
ELIB_SDFS_RETERR_MISC	Others														
Include file	srv_sdfs.h														
Calling sequence	int32_t Elib_SDFS_ChkDskAsync(UINT Ap_ID, int32_t Repair);														
Argument	Type	I/O	Description												
Ap_ID	UINT	I	Application ID												
Repair	int32_t	I	ELIB_SDFS_CHKDSK_ONLY (only check (without error correction)) ELIB_SDFS_CHKDSK_MODIFY (Detected error is corrected.)												
Return value	Type	-	Description												
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK: Normal end (processing start) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_WRITEPROTECT: Write												

			protect error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing)
Remark			

10.43 DPOF print target file count acquisition (synchronous)

Classification	SD management file service support function		
Function	DPOF print target file count acquisition	Symbol	Elib_SDFS_GetDPOFSetFileNum
Function overview	<p>This function obtains the number of images with print specified.</p> <ul style="list-style-type: none"> - This function obtains the number of images with DPOF set in an SD card. - The caller should allocate a field for the count argument. - This function operates synchronously. 		
Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_GetDPOFSetFileNum(UINT Ap_ID, uint16_t *count);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
count	uint16_t*	O	Number of images with DPOF set (The caller should allocate the field.)
Return value	Type	-	Description
Ret	int32_t	O	<ul style="list-style-type: none"> - For normal end ELIB_SDFS_RETOK - For abnormal end (negative numeric) ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_MISC: Other errors
Remark			

10.44 DPOF print target page count acquisition (asynchronous)

Classification	SD management file service support function																
Function	DPOF print target page count acquisition	Symbol	Elib_SDFS_GetDPOFCountAsync														
Function overview	This function obtains the number of pages to be printed with DPOF set. - This function obtains the number of pages to be printed with DPOF set in a specified static image file. - This function operates asynchronously. - No event occurs unless this function terminates normally (processing starts) (return value: ELIB_SDFS_RETOK).																
After terminating acquisition, this function creates and sends events listed below (info, subinfo = additional information in event transmission data).																	
<table><tr><th>Event symbol</th><th>info (transmission data)</th><th>subinfo (transmission data)</th><th>Explanation</th></tr><tr><td>SDFSNotify_DPOF_Count_GET_END</td><td>SDFSNotify_DPOF_Count_GET_END</td><td>ELIB_SDFS_RETOK</td><td>Normal end</td></tr><tr><td>SDFSNotify_DPOF_Count_GET_END</td><td>SDFSNotify_DPOF_Count_GET_END</td><td>Error type (negative)</td><td>Abnormal end</td></tr></table>				Event symbol	info (transmission data)	subinfo (transmission data)	Explanation	SDFSNotify_DPOF_Count_GET_END	SDFSNotify_DPOF_Count_GET_END	ELIB_SDFS_RETOK	Normal end	SDFSNotify_DPOF_Count_GET_END	SDFSNotify_DPOF_Count_GET_END	Error type (negative)	Abnormal end		
Event symbol	info (transmission data)	subinfo (transmission data)	Explanation														
SDFSNotify_DPOF_Count_GET_END	SDFSNotify_DPOF_Count_GET_END	ELIB_SDFS_RETOK	Normal end														
SDFSNotify_DPOF_Count_GET_END	SDFSNotify_DPOF_Count_GET_END	Error type (negative)	Abnormal end														
- Error types are listed below.																	
<table><tr><th>Symbol</th><th>Type</th></tr><tr><td>ELIB_SDFS_RETERR_NODIR</td><td>No directory</td></tr><tr><td>ELIB_SDFS_RETERR_NOFILE</td><td>No file</td></tr><tr><td>ELIB_SDFS_RETERR_ILLEGAL_PARAM</td><td>Parameter error</td></tr><tr><td>ELIB_SDFS_RETERR_IO</td><td>I-O error</td></tr><tr><td>ELIB_SDFS_RETERR_ABORTCALLED</td><td>Halt with halt request</td></tr><tr><td>ELIB_SDFS_RETERR_MISC</td><td>Others</td></tr></table>				Symbol	Type	ELIB_SDFS_RETERR_NODIR	No directory	ELIB_SDFS_RETERR_NOFILE	No file	ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error	ELIB_SDFS_RETERR_IO	I-O error	ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request	ELIB_SDFS_RETERR_MISC	Others
Symbol	Type																
ELIB_SDFS_RETERR_NODIR	No directory																
ELIB_SDFS_RETERR_NOFILE	No file																
ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error																
ELIB_SDFS_RETERR_IO	I-O error																
ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request																
ELIB_SDFS_RETERR_MISC	Others																
The data below is passed as structure of data sent via the event transmitter (see Section 4.4). uint16_t no; /* number of printed pages */																	
Include file	srv_sdfs.h																
Calling sequence	int32_t Elib_SDFS_GetDPOFCountAsync(UINT Ap_ID, uint16_t DataID);																
Argument	Type	I/O	Description														
Ap_ID	UINT	I	Application ID														
DataID	uint16_t	I	This argument specifies the data ID of a static image file.														
Return value	Type	-	Description														
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK: Normal end - For abnormal end (negative numeric)														

		ELIB_SDFS_RETERR_NODIR: File not found ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing)
Remark		

10.45 DPOF print target page count set (asynchronous)

Classification	SD management file service support function		
Function	DPOF print target page count set	Symbol	Elib_SDFS_SetDPOFCountAsync
Function overview	This function sets the number of pages to be printed for a specified file.		

- This function sets the number of pages to be printed for a specified static image file.
- This function operates asynchronously.
- No event occurs unless this function terminates normally (processing starts) (return value: ELIB_SDFS_RETOK).

After terminating set processing, this function creates and sends events listed below (info, subinfo = additional information in event transmission data).

Event symbol	info (transmission data)	subinfo (transmission data)	Explanation
SDFSNotify_DPOF_Count_SET_END	SDFSNotify_DPOF_Count_SET_END	ELIB_SDFS_RETOK	Normal end
SDFSNotify_DPOF_Count_SET_END	SDFSNotify_DPOF_Count_SET_END	Error type (negative)	Abnormal end

- Error types are listed below.

Symbol	Type
ELIB_SDFS_RETERR_NODIR	No directory
ELIB_SDFS_RETERR_NOFILE	No file
ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error
ELIB_SDFS_RETERR_IO	I-O error
ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request
ELIB_SDFS_RETERR_MEMFULL	Memory filled
ELIB_SDFS_RETERR_MISC	Others

Data is set in structure shown below.

	DPOF print target page count set structure	Tag name	ELIB_SDFS_DPOFPRINT
<pre> Typedef struct elib_sdfs_dpofprint { uint16_t DataID /* static image file data ID */ uint16_t Num; /* number of pages to be printed (0 to 99) */ uint32_t Format; /* image format */ } ELIB_SDFS_DPOFPRINT; </pre>			

Image formats are listed below.

ELIB_SDFS_DPOF_CIFF1	CIFF format	(DPOF print format)
ELIB_SDFS_DPOF_EXIF1_T	EXIFver1 and TIFF	(DPOF print format)
ELIB_SDFS_DPOF_EXIF1_J	EXIFver1 and JPEG	(DPOF print format)
ELIB_SDFS_DPOF_EXIF2_T	EXIFver2 and TIFF	(DPOF print format)
ELIB_SDFS_DPOF_EXIF2_J	EXIFver2 and JPEG	(DPOF print format)
ELIB_SDFS_DPOF_JFIF	JFIF format	(DPOF print format)
ELIB_SDFS_DPOF_FPX1	FPX1?	(DPOF print format)
ELIB_SDFS_DPOF_UNDEF	Undefined	(DPOF print format)

The data below is passed as structure of data sent via the event transmitter (see Section 4.4).

```
uint16_t no ; /* number of files with setting completed */
```

Include file	srv_sdfs.h		
Calling sequence	int32_t Elib_SDFS_SetDPOFCountAsync(UINT Ap_ID, ELIB_SDFS_DPOFPRINT *Dpofprint, uint16_t Count);		
Argument	Type	I/O	Description
Ap_ID	UINT	I	Application ID
Dpofprint	ELIB_SDFS_DPOFPRINT*	I	This argument indicates the pointer to DPOF print target page count set structure (The high-order process should allocate an area for the structure.)
Count	uint16_t	I	This argument specifies the number of files.
Return value	Type	-	Description
Ret	int32_t	O	- For normal end ELIB_SDFS_RETOK: Normal end (halt request accepted) - For abnormal end (negative numeric) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD management file service processing) ELIB_SDFS_RETERR_WRITEPROTECT: Write protect error
Remark			

10.46 DPOF target file deletion (asynchronous)

Classification	SD management file service support function																
Function	DPOF target file deletion	Symbol	Elib_SDFS_DPOFClearAsync														
Function overview	This function deletes a file with DPOF set.																
<div>- This function operates asynchronously.</div> <div>- No event occurs unless this function terminates normally (processing starts) (return value: ELIB_SDFS_RETOK).</div>																	
After terminating import, this function creates and sends events listed below (info, subinfo = additional information in event transmission data).																	
<table><tr><th>Event symbol</th><th>info (transmission data)</th><th>subinfo (transmission data)</th><th>Explanation</th></tr><tr><td>SDFSNotify_INFOSET_END</td><td>SDFSNotify_INFOSET_END</td><td>ELIB_SDFS_RETOK</td><td>Normal end</td></tr><tr><td>SDFSNotify_INFOSET_END</td><td>SDFSNotify_INFOSET_END</td><td>Error type (negative)</td><td>Abnormal end</td></tr></table>				Event symbol	info (transmission data)	subinfo (transmission data)	Explanation	SDFSNotify_INFOSET_END	SDFSNotify_INFOSET_END	ELIB_SDFS_RETOK	Normal end	SDFSNotify_INFOSET_END	SDFSNotify_INFOSET_END	Error type (negative)	Abnormal end		
Event symbol	info (transmission data)	subinfo (transmission data)	Explanation														
SDFSNotify_INFOSET_END	SDFSNotify_INFOSET_END	ELIB_SDFS_RETOK	Normal end														
SDFSNotify_INFOSET_END	SDFSNotify_INFOSET_END	Error type (negative)	Abnormal end														
<div>- Error types are listed below.</div> <table><tr><th>Symbol</th><th>Type</th></tr><tr><td>ELIB_SDFS_RETERR_NODIR</td><td>No directory</td></tr><tr><td>ELIB_SDFS_RETERR_NOFILE</td><td>No file</td></tr><tr><td>ELIB_SDFS_RETERR_ILLEGAL_PARAM</td><td>Parameter error</td></tr><tr><td>ELIB_SDFS_RETERR_IO</td><td>I-O error</td></tr><tr><td>ELIB_SDFS_RETERR_ABORTCALLED</td><td>Halt with halt request</td></tr><tr><td>ELIB_SDFS_RETERR_MISC</td><td>Others</td></tr></table>				Symbol	Type	ELIB_SDFS_RETERR_NODIR	No directory	ELIB_SDFS_RETERR_NOFILE	No file	ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error	ELIB_SDFS_RETERR_IO	I-O error	ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request	ELIB_SDFS_RETERR_MISC	Others
Symbol	Type																
ELIB_SDFS_RETERR_NODIR	No directory																
ELIB_SDFS_RETERR_NOFILE	No file																
ELIB_SDFS_RETERR_ILLEGAL_PARAM	Parameter error																
ELIB_SDFS_RETERR_IO	I-O error																
ELIB_SDFS_RETERR_ABORTCALLED	Halt with halt request																
ELIB_SDFS_RETERR_MISC	Others																
Include file	srv_sdfs.h																
Calling sequence	int32_t Elib_SDFS_DPOFClearAsync(UINT Ap_ID);																
Argument	Type	I/O	Description														
Ap_ID	UINT	I	Application ID														
Return value	Type	-	Description														
Ret	int32_t	O	<div>- For normal end ELIB_SDFS_RETOK: Normal end (halt request accepted)</div> <div>- For abnormal end (negative numeric) ELIB_SDFS_RETERR_ILLEGAL_PARAM: Parameter error ELIB_SDFS_RETERR_IO: I-O error ELIB_SDFS_RETERR_MISC: Other errors ELIB_SDFS_BLOCKING: Blocking error (during SD</div>														

		management file service processing) ELIB_SDFS_RETERR_WRITEPROTECT: protect error	Write
Remark			

10.47 File open

Classification	SD standard file system function		
Function	File open	Symbol	FS_fopen_SD
Overview	<p>This function opens a file specified with a file name in argument “filename” by using an access mode specified in argument “mode”.</p> <p>If an error occurs in file open, null is returned. To check the error, the error acquisition function need be used.</p> <p>For example, incorrect card read and incorrect format are handled as SD card errors.</p> <p>Whether an SD card can be accessed can be recognized at the first file open after the SD card is inserted.</p> <p>The value of structure internal variable T_MSL_SDFIL_FILE returned from this function must not be referenced/changed with EILB(APL). The file pointer variable is used with other I-O functions explained below.</p>		
Include file	#include "msl_sdfil.h"		
Calling sequence	T_MSL_SDFIL_FILE *FS_fopen_SD(const int8_t *filename ,const int8_t *mode);		
Argument	Type	I/O	Description
Filename	const int8_t *	I	This argument specifies a file name.
Mode	const int8_t *	I	This argument specifies usable mode (see the remark for details).
Return value	T_MSL_SDFIL_FILE *		Normal end: file pointer (!NULL) Abnormal end: NULL
Remark			

- Access mode

Descriptor	Access mode
r	Read mode (text mode) is used. If a specified file is not present or not found, an error occurs.
w	Write mode (text mode) is used. If a specified file is already allocated, data in it is discarded.
a	Write mode (text mode) is used. If a specified file is already allocated, data is added to the end of it.
rb	"r" in binary mode
wb	"w" in binary mode
ab	"a" in binary mode
r+	Read mode and write mode are used. If a specified file is not found, an error occurs.
w+	Read mode and write mode are used. If a specified file is already allocated, data in it is discarded.
a+	Read mode and write mode are used. If a specified file is already allocated, data is added to the end of it.
r+b	"r+" in binary mode
w+b	"w+" in binary mode
a+b	"a+" in binary mode
d	Open directory

* Note

When text mode is used, the carriage return & line feed sequence (CR-LF) sequence is replaced with line feed (LF) at file access.

This applies to file open in mode without "b" in its descriptor (e.g., "r", "w", "a").

- Use example

```
T_MUS_SDFIL_FILE *fp;
```

```
fp = FS_fopen_SD( "/mnt/sdcard/temp" , "r+b" );  
if(fp== NULL)  
{  
  //unsuccessful file open  
  errcode=FS_ferror_SD(NULL);//error code acquisition  
}
```

10.48 File closing

Classification	SD standard file system function		
Function	File closing	Symbol	FS_fclose_SD
Overview	<p>This function closes an opened file indicated by a file pointer specified in argument “fp”. If data has been written in the specified file, data in the output buffer is written in the physical file and the buffer area is released.</p>		
Include file	#include "msl_sdfil.h"		
Calling sequence	int32_t FS_fclose_SD(T_MSL_SDFIL_FILE *fp);		
Argument	Type	I/O	Description
fp	T_MSL_SDFIL_FILE *	I	This argument indicates the pointer to structure T_MSL_SDFIL_FILE.
Return value	int32_t		Normal end: D_MSL_SDFIL_OK (0) Abnormal end: Error code (negative numeric)
Remark	<p>- Use example</p> <pre>if((errcode=FS_fclose_SD(fp))<0) { //unsuccessful closing and recovery or debugging }</pre>		

10.49 File read

Classification	SD standard file system function		
Function	File read	Symbol	FS_fread_SD
Overview	<p>This function reads data containing items as many as a value specified in argument “count” with an item size (number of bytes) specified in “size” from an opened file indicated by a file pointer specified in “fp” and stores it in an area specified in “buffer”.</p> <p>A data type can be specified freely in the “buffer” argument.</p> <p>This function returns the number of read items. If this return value is smaller than the value specified in the “count” argument, an error occurs in the file read or the total number of data items in the file is smaller than the “count” value.</p>		
Include file	#include "msl_sdfil.h"		
Calling sequence	int32_t FS_fread_SD(void *buffer ,int32_t size , int32_t count ,T_MSL_SDFIL_FILE *fp);		
Argument	Type	I/O	Description
buffer	void *	O	This argument specifies the pointer to the top of an area to store read data.
size	int32_t	I	This argument specifies a data item size (1 is recommended).
count	int32_t	I	This argument specifies the maximum number of items to be read (or number of bytes to be read when 1 is set in argument “size”).
fp	T_MSL_SDFIL_FILE *	I	This argument indicates the pointer to structure T_MSL_SDFIL_FILE.
Return value	int32_t		Normal end: Number of read items (0 or positive numeric) Abnormal end: Error code (negative numeric)
Remark			

* Notes

1. When the target file is opened in text mode, the carriage return & line feed sequence (CR-LF) sequence is replaced with line feed (LF).
2. This function returns the number of read data items and so, if a value not less than 2 is specified as a data item size and the target file size is not a multiple of the data item size, the number of actually read bytes cannot be recognized. So, set 1 as a data item size as a rule.

- Use example

```
if( (errcode=FS_fread_SD(buf, 1, 80, fp))<0 )  
{  
    //unsuccessful read and recovery or debugging  
}
```

10.50 File writing

Classification	SD standard file system function		
Function	File writing	Symbol	FS_fwrite_SD
Overview	<p>This function reads data containing items as many as a value specified in argument “count” with an item size specified in “size” from an area specified in “buffer” and writes it in an opened file indicated by a file pointer specified in “fp”.</p> <p>A data type can be specified freely in the “buffer” argument.</p> <p>This function returns the number of written items. If this return value is smaller than the value specified in the “count” argument, a write error occurs (or the total size of items to be written exceeds the target SD size). In other words, if the return value differs from the specified “count” value, an error occurs (because it is assumed that this function is called with the target SD card size recognized).</p>		
Include file	#include "msl_sdfil.h"		
Calling sequence	int32_t FS_fwrite_SD(const void *buffer , int32_t size , int32_t count ,T_MSL_SDFIL_FILE *fp);		
Argument	Type	I/O	Description
buffer	const void *		This argument specifies the pointer to the top of an area storing data to be written.
size	int32_t		This argument specifies a data item size.
count	int32_t		This argument specifies the number of items to be written.
fp	T_MSL_SDFIL_FILE *		This argument indicates the pointer to structure T_MSL_SDFIL_FILE.
Return value	int32_t		Normal end: Number of written items (0 or positive numeric) Abnormal end: Error code (negative numeric)
Remark	<p>* Note</p> <p>When the target file is opened in text mode, line feed (LF) is replaced with the carriage return & line feed sequence (CR-LF) sequence.</p> <p>- Use example</p> <pre>if((errcode=FS_fwrite_SD(data, sizeof(int), 10, fp)) < 0) { //unsuccessful writing and recovery or debugging }</pre>		

10.51 Seek

Classification	SD standard file system function		
Function	Seek	Symbol	FS_fseek_SD
Overview	<p>This function moves the access position (file position specifier) of an opened file indicated by a file pointer specified in “fp”.</p> <p>The access position is moved to a relative position with a position specified in argument “origin” used as origin. It is moved forward when a positive value is specified in argument “offset” or backward when a negative value is specified in the argument.</p>		
Include file	#include "msl_sdfil.h"		
Calling sequence	int32_t FS_fseek_SD(T_MSL_SDFIL_FILE * fp , long offset , int32_t origin);		
Argument	Type	I/O	Description
fp	T_MSL_SDFIL_FILE *	I	This argument indicates the pointer to structure T_MSL_SDFIL_FILE.
offset	long	I	This argument specifies the number of bytes by which the target is to be moved from the “origin” value.
origin	int32_t	I	This argument specifies the initial file pointer value as follows: D_MSL_SDFIL_SEEK_SET; § File top D_MSL_SDFIL_SEEK_CUR; i § Current file pointer value D_MSL_SDFIL_SEEK_END; § File end
Return value	int32_t		Normal end: D_MSL_SDFIL_OK (0) Abnormal end: Error code (negative numeric)
Remark	<p>- Use example</p> <pre>if((errcode=FS_fseek_SD(fp, 10,SEEK_SET))<0) { //unsuccessful movement and recovery or debugging }</pre>		

10.52 Total free area size (byte) acquisition

Classification	SD standard file system function
----------------	----------------------------------

Function	Total free area size (byte) acquisition	Symbol	FS_free_byte_cnt_SD
Overview	<p>This function obtains the total free area size (number of bytes) in an SD card.</p>		
Include file	#include "msl_sdfile.h"		
Calling sequence	int32_t FS_free_byte_cnt_SD(void);		
Argument	Type	I/O	Description
Return value	int32_t		Normal end: Value not less than 0 (obtained size) Abnormal end: Error code (negative numeric)
Remark	<p>- Use example</p> <pre> if((freebytes = FS_free_byte_cnt_SD())<0) { //unsuccessful acquisition and recovery or debugging } </pre>		

10.53 File deletion

Classification	SD standard file system function		
Function	File deletion	Symbol	FS_remove_SD
Overview	<p>This function deletes a file with a name specified in argument “pathname”.</p> <p>Specify a full pathname with a drive name in the “pathname” argument.</p> <p>If a directory name is specified in the “pathname” argument, the target cannot be deleted.</p>		
Include file	#include "msl_sdfil.h"		
Calling sequence	int32_t FS_remove_SD(const int8_t *pathname);		
Argument	Type	I/O	Description
pathname	const int8_t *	I	This argument specifies the name (full pathname with drive name) of a file to be deleted.
Return value	int32_t		Normal end: D_MSL_SDFIL_OK (0) Abnormal end: Error code (negative numeric)
Remark	<p>- Use example</p> <pre>if((FS_remove_SD(fp))<0) { //unsuccessful deletion and debugging }</pre>		

10.54 File information acquisition

Classification	SD standard file system function		
Function	File information acquisition	Symbol	FS_readfileinfo_SD
Overview	<p>This function obtains information about a file specified in argument “pathname”.</p> <p>Specify the number of pathname arrays (information items to be obtained) in argument “cnt”.</p> <p>Obtained information is stored in structure specified in argument “dirp”. The caller should allocate structure area.</p>		
Include file	#include "msl_sdfil.h"		
Calling sequence	<pre>int32_t FS_readfileinfo_SD(T_MSL_SDFIL_DIRENT *dirp ,const int8_t *pathname[] , int32_t cnt);</pre>		
Argument	Type	I/O	Description
dirp	T_MSL_SDFIL_DIRENT *		This argument specifies structure to store obtained file information. The caller should allocate fields (as many as the number of information items to be obtained) of the structure to store the information.
pathname	const int8_t (*)[]		This argument specifies the pointer to a file path name array.
cnt	int32_t		Number of information items to be obtained.
Return value	int32_t		Normal end: Positive value (number of obtained information items) Abnormal end: Error code (negative numeric)
Remark	<p>* The name of a target file cannot be obtained from the file system and so, a character string between the last \$ and \$0 after it in the file path name set in the “pathname” argument is stored as file name (directory name). If the character string ends with \$, this \$ is deleted before storage.</p> <p>* See Section 3-1, “T_MSL_SDFIL_DIRENT structure”.</p>		

10.55 SD general area format check

Classification	Extended function for SD control		
Function	SD general area format check	Symbol	FS_formatcheck_SD
Overview	<p>This function checks the format of an SD card (synchronously).</p>		
Include file	#include "msl_sdfil.h"		
Calling sequence	int32_t FS_formatcheck_SD(const int8_t *drivename , int32_t checktype);		
Argument	Type	I/O	Description
drive	const int8_t *	I	This argument specifies the pointer to a target drive name.
checktype	int32_t	I	This argument specifies a check type (0: only check, 1: check and correction) D_MSL_SDFIL_CHKDSKONRY : 0 D_MSL_SDFIL_CHKDSKFIX : 1
Return value	int32_t		Normal end : D_MSL_SDFIL_OK (0) (no error detected) Abnormal end: Negative numeric
Remark			

10.56 File error acquisition

Classification	Error function		
Function	File error acquisition	Symbol	FS_ferror_SD
Overview	<p>This function obtains errors in file access for each file pointer. It also obtains errors occurring with a file pointer specified in argument “fp”.</p>		
Include file	#include "msl_sdfil.h"		
Calling sequence	int32_t FS_ferror_SD(T_MSL_SDFIL_FILE *fp);		
Argument	Type	I/O	Description
fp	T_MSL_SDFIL_FILE *	I	This argument indicates the pointer to structure T_MSL_SDFIL_FILE structure. For null, error value in the preceding open
Return value	int32_t		No error: D_MSL_SDFIL_OK (0) Abnormal end: Error code (negative numeric)
Remark			

11. Resource Management Library Interface

11.1 Resetting User-defined Items

Classification	Initialization processing		
Function name	Resetting user-defined items	Symbol	Msl_OpeUsrSetRset
Function overview	<p>This function is used to initialize user-defined items.</p>		
Include file	Res_ope.h		
Calling sequence	Int Msl_OpeUsrSetRset(mode);		
Argument name	Type	I/O	Explanation
Mode	int	I	<p>Specifies initialization mode.</p> <p>MSL_URSET_NORM: Initialization by the user</p> <p>MSL_URSET_D1: Initialization by the D1 command.</p> <p>MSL_URSET_EEPROM: Initializes the EEPROM area.</p> <p>MSL_URSET_EEPROM_AC: Initializes the EEPROM area (A + C).</p> <p>MSL_URSET_USERDATA: Initializes user data.</p>
Return value	Type	I/O	Explanation
Ret	int	O	<p>Processing result</p> <p>MSL_OK: Normal end</p> <p>MSL_NG: Abnormal end (*1)</p>
Remarks	<p>*1 Initialization processing result NG</p>		

11.2 Setting or Referencing Communication Parameters

Classification	Terminal setting		
Function name	Setting or referencing communication parameters	Symbol	Msl_OpeCrspParaSetRef
Function overview	<p>This function is used to set or reference the communication parameter values. (Values are shown below.)</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeCrspParaSetRef(mode, para, buff) ;		
Argument name	Type	I/O	Explanation
Mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
Para	int	I	Communication parameter type (*1)
Buff	unsigned char *	I	Setting or reference value storage area
Return value	Type	I/O	Explanation
Ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks	<p>Data is stored in the setting or reference value storage area, in units of bytes.</p> <p>*1 Specify the following define values for the communication parameter type:</p> <pre> #define MSL_HTTPFLOWPOINT /* Setting of HTTP flow point output */ #define MSL_WCDMAFLOWPOINT /* Setting of TCP or WCDMA flow point output */ #define MSL_HTTPCONTOUT /* Setting of HTTP signal content output */ #define MSL_WCDMACONTOUT /* Setting of TCP or WCDMA signal content output */ #define MSL_HTTPTIMER1 /* HTTP timer 1 */ #define MSL_HTTPTIMER2 /* HTTP timer 2 */ #define MSL_SYNTOUT /* Setting of SYN timeout value */ #define MSL_RTTINIT /* Setting of Initial RTT */ #define MSL_TCPRETRANINTERVALMAX /* Setting of TCP retransmission interval minimum value */ </pre>		


```
#define MSL_TCPRETRANINTERVALMIN /* Setting of TCP retransmission interval  
maximum value */  
#define MSL_MAXRETRAN_NUM /* Maximum retransmission count */  
#define MSL_DELAYACKTOUT /* Setting of delay ACK timeout value */  
#define MSL_REFORWARD_BORDER /* Setting of high speed retransfer  
threshold */  
#define MSL_RECWINSZ /* Setting of reception window size */  
#define MSL_TRANWINSZ /* Setting of transmission window size */  
#define MSL_SOCKETMAX /* Setting of maximum number of  
sockets */
```

11.3 Setting or Referencing Functions

Classification	Terminal setting		
Function name	Setting or referencing functions	Symbol	Msl_OpeSetRef
Function overview	<p>This function is used to set or reference each function.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeSetRef(mode, no, data) ;		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
no	unsigned char	I	Specifies a function number.
data	unsigned char	I	Specifies a setting value.
Return value	Type	I/O	Explanation
ret	int	O	Processing result Return value of the internal setting function: Normal end MSL_NG: Abnormal end (The function number is less than MSL_DUMMY.)
Remarks			

11.4 Setting or Referencing Tone Gain Specification (CDTNG or CDTONE) 2

Classification	Terminal setting		
Function name	Setting or referencing tone gain specification (CDTNG or CDTONE) 2	Symbol	Msl_OpeToneGainSetRef2
Function overview	<p>This function is used to set or reference the tone gain specification (CDTNG or CDTONE).</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeToneGainSetRef2 (mode, tmode, data) ;		
Argument name	Type	I/O	Explanation
Mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
tmode	int	I	Tone gain mode MSL_TONE1GAIN: Tone 1 gain specification (CDTNG1) MSL_TONE2GAIN: Tone 2 gain specification (CDTNG2) MSL_TONE3GAIN: Tone 3 gain specification (CDTNG3) MSL_TONE1GAIN_HF_SET: Tone gain 1 (handsfree) level MSL_TONE1GAIN_HF_TALKVOL_SET: Tone gain 1 (receiving volume changing tone for handsfree use) level MSL_TONE1GAIN_HF_ALARM_SET: Tone gain 1 (product quality degradation alarm for handsfree use) MSL_TONE3GAIN_HF_SET: Tone gain 3 (for handsfree use) level MSL_TONE3GAIN_HF_TALKVOL_SET: Tone gain 3 (receiving volume changing tone for handsfree use) level MSL_TONE3GAIN_HF_ALARM_SET: Tone gain 3 (product quality degradation alarm for handsfree use)

			MSL_ATT_HF_SET: Tone ATT (for handsfree use)
data	unsigned char *	I/O	Pointer of the tone gain specification (CDTNG) storage area
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks	<p><Details of data></p> <ul style="list-style-type: none"> -Tone 1 gain specification (CDTNG1): 6 bytes <ul style="list-style-type: none"> Byte 1: Tone gain 1 (key tone for mainframe standby) Byte 2: Tone gain 1 (for earphone use) Byte 3: Tone gain 1 (for mainframe use) Byte 4: Tone gain 1 (for USB handsfree use) Byte 5: Tone gain 1 (on-hold tone) Byte 6: Tone gain 1 (product quality degradation alarm) - Tone 2 gain specification (CDTNG2): 4 bytes <ul style="list-style-type: none"> Byte 1: Tone gain 2 (DTMF for mainframe standby) Byte 2: Tone gain 2 (DTMF for mainframe communication) Byte 3: Tone gain 2 (DTMF for earphone use) Byte 4: Tone gain 2 (DTMF for USB handsfree use) - Tone 3 gain specification (CDTNG3): 6 bytes <ul style="list-style-type: none"> Byte 1: Tone gain 3 (key tone for mainframe standby) Byte 2: Tone gain 3 (key tone for mainframe communication) Byte 3: Tone gain 3 (key tone for earphone use) Byte 4: Tone gain 3 (key tone for USB handsfree use) Byte 5: Tone gain 3 (product quality degradation alarm for mainframe or earphone use) Byte 6: Tone gain 3 (on-hold tone) - Tone gain 1 (for handsfree use) level: 3 bytes <ul style="list-style-type: none"> Byte 1: Tone gain 1 (for handsfree use) level 1 or 4 Byte 2: Tone gain 1 (for handsfree use) level 2 or 5 Byte 3: Tone gain 1 (for handsfree use) level 3 or 6 - Tone gain 1 (receiving volume changing tone for handsfree use) level: 3 bytes <ul style="list-style-type: none"> Byte 1: Tone gain 1 (receiving volume changing tone for handsfree use) level 1 or 4 Byte 2: Tone gain 1 (receiving volume changing tone for handsfree use) level 2 or 5 Byte 3: Tone gain 1 (receiving volume changing tone for handsfree use) level 3 or 6 - Tone gain 1 (product quality degradation alarm): 3 bytes 		

- Byte 1: Tone gain 1 (product quality degradation alarm for handsfree use) level 1 or 4
- Byte 2: Tone gain 1 (product quality degradation alarm for handsfree use) level 2 or 5
- Byte 3: Tone gain 1 (product quality degradation alarm for handsfree use) level 3 or 6
- Tone gain 3 (for handsfree use) level: 4 bytes
 - Byte 1: Tone gain 3 (for handsfree use) level 1 or 2
 - Byte 2: Tone gain 3 (for handsfree use) level 3
 - Byte 3: Tone gain 3 (for handsfree use) level 4 or 5
 - Byte 4: Tone gain 3 (for handsfree use) level 6
- Tone gain 3 (receiving volume changing tone for handsfree use) level: 4 bytes
 - Byte 1: Tone gain 3 (receiving volume changing tone for handsfree use) level 1 or 2
 - Byte 2: Tone gain 3 (receiving volume changing tone for handsfree use) level 3
 - Byte 3: Tone gain 3 (receiving volume changing sound for handsfree use) level 4 or 5
 - Byte 4: Tone gain 3 (receiving volume changing sound for handsfree use) level 6
- Tone gain 3 (product quality degradation alarm for handsfree use) level: 4 bytes
 - Byte 1: Tone gain 3 (product quality degradation alarm for handsfree use) level 1 or 2
 - Byte 2: Tone gain 3 (product quality degradation alarm for handsfree use) level 3
 - Byte 3: Tone gain 3 (product quality degradation alarm for handsfree use) level 4 or 5
 - Byte 4: Tone gain 3 (product quality degradation alarm for handsfree use) level 6
- Tone ATT specification: 3 bytes
 - Byte 1: Tone ATT (for handsfree use)
 - Byte 2: Tone ATT (receiving volume changing tone for handsfree use)
 - Byte 3: Tone ATT (product quality degradation alarm for handsfree use)

11.5 Setting or Referencing Wakeup Message

Classification	Message function		
Function name	Setting or referencing wakeup message	Symbol	Msl_OpeWakeupMsgSetRef
Function overview	<p>This function is used to set or reference the wakeup message and wakeup message size.</p> <ul style="list-style-type: none"> - Up to 100 bytes of wakeup messages can be used. 		
Include file	res_ope.h		
Calling sequence	int Msl_OpeWakeupMsgSetRef(mode, kind, len, wakeup_msg) ;		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
kind	unsigned char	I	Message type: MSL_WAKEUP_MSG (can only be specified)
len	unsigned char *	I/O	Wakeup message size:
wakeup_msg	unsigned char *	I/O	Wakeup message area:
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks	<p>Wakeup message size (maximum): unsigned char wakeup_msg[MSL_WAKEUPMSG_MAX+1]</p> <pre>#define MSL_WAKEUPMSG_MAX 100 /* Wakeup message size (maximum) */</pre>		

11.6 Setting or Referencing Clock Display

Classification	Clock function		
Function name	Setting or referencing clock display	Symbol	Msl_OpeClockDspSetRef
Function overview	<p>This function is used to register or reference the clock display settings.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeClockDspSetRef(mode, msl_clkdsp);		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
msl_clkdsp	_MSL_CLKDSP *	I/O	Clock display setting structure pointer. (See Remarks.)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks	<pre> [Clock display setting structure] typedef struct tagMSL_CLKDSP { unsigned char Dsp ; /* Display on or off */ unsigned char DspMode ; /* Japanese 24h (MSL_DATEMODE1) */ /* English 24h (MSL_DATEMODE2) */ /* Japanese 12h (MSL_DATEMODE3) */ /* English 12h (MSL_DATEMODE4) */ /* Analog (MSL_DATEMODE5) */ unsigned char Size ; /* Display big (MSL_CLKDSP_BIG) */ /* Display small (MSL_CLKDSP_SMALL) */ } _MSL_CLKDSP #define MSL_DATEMODE1 /* Japanese 24h */ #define MSL_DATEMODE2 /* English 24h */ #define MSL_DATEMODE3 /* Japanese 12h(AM/PM) */ #define MSL_DATEMODE4 /* English 12h(AM/PM) */ #define MSL_DATEMODE5 /* Analog */ </pre>		

```
#define MSL_CLKDSP_BIG      /* Display big      */  
#define MSL_CLKDSP_SMALL  /* Display small  */
```


11.7 Setting or Referencing Formatted Text Folder Name

Classification	General function		
Function name	Setting or referencing formatted text folder name	Symbol	Msl_OpePresetFldNameSetRef
Function overview	<p>This function is used to set the name of a formatted text folder.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpePresetFldNameSetRef(mode, no, fldname) ;		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
no	int	I	Folder number (Folder number: 1 to 5)
fldname	_MSL_TEIKEI_NAME *	I/O	Folder name structure pointer
Return value	Type	I/O	Explanation
Ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (out of range of folder numbers)
Remarks	<pre>[Folder name structure] typedef struct tagMSL_TEIKEI_NAME { unsigned char Jpn_fldname[MSL_FLDNAMEMAX +1]; /* Japanese folder name */ unsigned char Eng_fldname[MSL_FLDNAMEMAX +1]; /* English folder name */ } _MSL_TEIKEI_NAME ; #define MSL_FLDNAMEMAX 20 /* Maximum size for the formatted text folder name */</pre>		

11.8 Setting Original Menu Number

Classification	General function		
Function name	Setting original menu number	Symbol	Msl_OpeOrgMenuSetRef
Function overview	<p>This function is used to set or reference the user-defined original menu. Up to 10 menu items can be set (0 to 9).</p> <p>Setting data must have a sub item number, ranging from 0 to 0x0fff.</p> <ul style="list-style-type: none"> - 0x0fff indicates no setting. 		
Include file	res_ope.h		
Calling sequence	int Msl_OpeOrgMenuSetRef(mode, num, menu) ;		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
num	unsigned char	I	Menu number: A value from 0 to MSL_MENUMAX-1 can be specified.
menu	unsigned short *	I/O	Sub item number: A value from 0 to 0x0fff can be specified.
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (out of range of setting data)
Remarks	<pre>#define MSL_MENUMAX 10 /* Maximum number of original menu items to be registered */</pre>		

11.9 Setting or Referencing Image Paste Destination Specification

Classification	General function		
Function name	Setting or referencing image paste destination specification	Symbol	Msl_OpePasteSetRef
Function overview	<p>This function is used to set or reference an original image with the image paste destination specified. (The values to be set or referenced are shown below.)</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpePasteSetRef(mode, num, data) ;		
Argument name	Type	I/O	Explanation
Mode	Int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
Num	Int	I	Specifies the paste destination screen. MSL_PASTE_TELSEND: Telephone outgoing screen MSL_PASTE_TELRCV: Telephone incoming screen MSL_PASTE_MAILSEND: Mail transmission screen MSL_PASTE_MAILRCV: Mail reception screen MSL_PASTE_TOIAWASE: Inquiry screen MSL_PASTE_WAITDSP: Standby screen MSL_PASTE_WAKEUP: Wakeup screen MSL_PASTE_BACKDSP: Rear LCD screen (standby) MSL_PASTE_BKTELRCV: Rear LCD screen (telephone incoming) MSL_PASTE_BKMAILRCV: Rear LCD screen (mail reception) (Not used) MSL_PASTE_WAITDSPCAL: Standby screen

			(Standby calendar background)
Data	_MSL_OPEIMGFOL *	I/O	Pointer of the setting or reference data structure Folder type MSL_OPE_FOLNON: No folder (not specified) MSL_OPE_FOL_BOX: InBox MSL_OPE_FOL_CAM: Camera folder (fixed) MSL_OPE_FOL_PRI: Preinstall MSL_OPE_FOL_ANI: Self-produced animation MSL_OPE_FOL_USR01: User folder 1 ~ MSL_OPE_FOL_USR20: User folder 20
Return value	Type	I/O	Explanation
Ret	Int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks	<p>[Setting or reference data structure]</p> <pre>typedef struct tagMSL_OPEIMGFOL { unsigned short data; /* Image number*/ unsigned char folder; /* Folder type*/ unsigned char Reserve; /* Reserved */ } _MSL_OPEIMGFOL ;</pre> <p>* Image number: To extend an original image, unsigned short must be used instead of unsigned char. Only the standby images can be used for i-Appli images.</p> <p>If, an original image number whose image is not registered is specified, operation after setting is not guaranteed. The upper-level application must determine whether the original image is registered before setting.</p> <p>Perform the following procedure to set a calendar (with or without a background</p>		

image) in the standby screen:

<Without a background image>

- (1) Use the following steps to specify a calendar in the standby screen:
 - Specify MSL_PASTE_WAITDSP (standby screen) in num (paste destination screen specification).
 - Specify MSL_OPE_FOL_PRI (preinstall) in data.folder (folder type).
 - Specify MSL_PASTE_CAL (calendar) in data.data (image number).

- (2) Then, use the following steps to specify no calendar background image (without) in the standby screen:

- Specify MSL_PASTE_WAITDSPCAL (rear LCD screen (standby calendar background)) in num (paste destination screen specification).
- Specify MSL_OPE_FOL_PRI (preinstall) in data.folder (folder type).
- Specify MSL_OPE_IMG_EMPTY (off) in data.data (image number).

<With a background image> Example: Set a self-produced animation.

- (1) Use the following steps to specify a calendar in the standby screen:
 - Specify MSL_PASTE_WAITDSP (standby screen) in num (paste destination screen specification).
 - Specify MSL_OPE_PRI (preinstall folder) in data.folder (folder type).
 - Specify MSL_PASTE_CAL (calendar) in data.data (image number).

- (2) Then, use the following steps to specify a calendar background image (with) in the standby screen:

- Specify MSL_PASTE_WAITDSPCAL (standby calendar background) in num (paste destination screen specification).
- Specify MSL_OPE_FOL_ANI (self-produced animation folder) in data.folder (folder type).
- Specify MSL_OPE_OWNaNIME01 (self-produced animation) in data.data (image number).

11.10 Setting or Referencing Pose Dial

Classification	General function		
Function name	Setting or referencing pose dial	Symbol	Msl_OpePauseDial
Function overview	<p>This function is used to set or reference a user-defined pose dial. Only one pose dial can be registered.</p> <ul style="list-style-type: none"> - The maximum size of the pose dial that can be set or referenced is 128 bytes. 		
Include file	res_ope.h		
Calling sequence	void Msl_OpePauseDial(mode, pause_dial);		
Argument name	Type	I/O	Explanation
mode	Int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
pause_dial	unsigned char *	I/O	Buffer for setting or referencing a pose dial (MAX 129 bytes) (including a null at the end)
Return value	Type	I/O	Explanation
-	void	I/O	-
Remarks	<p>Buffer for setting or referencing the maximum size of a pose dial: unsigned char pause_dial[MSL_PAUSEDIALMAX]</p> <pre>#define MSL_PAUSEDIALMAX 129 /* Pose dial data size (including a null at the end) */</pre>		

11.11 Setting or Referencing Original Manner Mode

Classification	General function		
Function name	Setting or referencing original manner mode	Symbol	Msl_OpeOrgMannerSetRef
Function overview	<p>This function is used to set or reference original manner mode.</p>		
Include file	res_ope.h		
Calling sequence	void Msl_OpeOrgMannerSetRef(mode, dat) ;		
Argument name	Type	I/O	Explanation
Mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
Dat	_MSL_OPEORGMANNER *	I/O	Data to be set or referenced
Return value	Type	I/O	Explanation
-	void	-	-
Remarks	<p>[Original manner mode structure]</p> <pre>typedef struct tagMSL_OPEORGMANNER { unsigned char dengon ; /* Message slip On or Off (default: Off) */ unsigned char den_time ; /* Call time setting */ unsigned char vibrator ; /* Vibrator On or Off (default: On) */ unsigned char tel_vol ; /* Telephone ring tone volume Mute/Levels 1 to 6/Step (default: Mute)(*1) */ unsigned char mail_vol ; /* Mail ring tone volume Mute/Levels 1 to 6/Step (default: Mute)(*1) */ unsigned char wakeup_vol ; /* Wakeup volume Mute/Levels 1 to 6/Step (default: Mute)(*1) */ unsigned char voice_mem ; /* Memo checking tone On or Off (default: On) */ unsigned char button; /* Button checking tone On or Off (default: Off) */ unsigned char tuuwamike ; /* Communicating microphone sensitivity Normal/Up (On or Off) (default: Up) */ unsigned char battalarm ; /* Low-voltage alarm tone On or Off (default:</pre>		

```

Off)          */
} _MSL_OPEORGMANNER ;

*1
#define MSL_SILENT          /* Mute          */
#define MSL_VOLUMELVL1      /* Volume level 1*/
#define MSL_VOLUMELVL2      /* Volume level 2*/
#define MSL_VOLUMELVL3      /* Volume level 3*/
#define MSL_VOLUMELVL4      /* Volume level 4*/
#define MSL_VOLUMELVL5      /* Volume level 5*/
#define MSL_VOLUMELVL6      /* Volume level 6*/
#define MSL_VOLUMESTEP      /* Step          */

```


11.12 Setting or Referencing Various Ring Tone Patterns

Classification	General function		
Function name	Setting or referencing various ring tone patterns	Symbol	Msl_OpeRcvToneSetRef
Function overview	<p>This function is used to set or reference various ring tone patterns. See "Setting or Referencing Various Ring Tone Patterns for Using TV Phone (Msl_OpeTVTel_RcvToneSetRef)," for details of setting or referencing TV phone ring tone patterns.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeRcvToneSetRef(mode, func_no, sound_id) ;		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
func_no	unsigned char	I	Function number MSL_RCVTONE_TEL: Normal ring tone MSL_RCVTONE_IMAIL: Ring tone for i-mail MSL_RCVTONE_MSGR: Ring tone for message R MSL_RCVTONE_MSGF: Ring tone for message F MSL_RCVTONE_IMODETEL: Ring tone for voice MSL_RCVTONE_DIAL_LIMIT: Ring tone for unnoticed call MSL_RCVTONE_PUBLIC_PHONE: Ring tone for public phone call MSL_RCVTONE_NOT_NOTICE: Ring tone for notice impossible
sound_id	short *	I/O	Sound ID
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (out of range of function numbers)

Remarks	
---------	--

11.13 Registering or Referencing Fixed Formatted Texts

Classification	General function		
Function name	Registering or referencing fixed formatted texts	Symbol	Msl_OpeSentenceSetRef
Function overview	<p>This function is used to read a fixed formatted text of the specified formatted text number.</p> <p>Formatted text numbers</p> <p>1 to 20: Fixed formatted text (20 types, search word, including kana formatted text)</p> <p>21 to 50: Free formatted text (30 types, no search word): Out of target</p> <p>If a formatted text number from 1 to 20 is specified, the structure received as an argument name is stored in NAND FLASH.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeSentenceSetRef(mode, no, param, data) ;		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
no	unsigned char	I	Formatted text number (1 to 20)
param	unsigned char	I	Read mode MSL_TEIKEI_ENGLISH: English MSL_TEIKEI_JAPAN: Japanese
data	_MSL_TEIDATA_KOTEI *	O	Fixed formatted text structure pointer
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			

The formatted texts for folders 1 to 5 are as follows:

- Folder 1: Formatted text numbers 1 to 10
- Folder 2: Formatted text numbers 11 to 20
- Folder 3: Formatted text numbers 21 to 30
- Folder 4: Formatted text numbers 31 to 40
- Folder 5: Formatted text numbers 41 to 50

[Fixed formatted text structure]

```
typedef struct tagMSL_TEIDATA_KOTEI
{
    unsigned char    entry_flg ;           /* Flag indicating with or without user
registration          */
    unsigned char    tei_eng[MSL_TEIKEI_MAX+1]; /* Fixed formatted text:
English              */
    unsigned char    tei_kan[MSL_TEIKEI_MAX+1]; /* Fixed formatted text:
kanji                */
    unsigned char    kana[MSL_TEIKEI_KANAMAX+1]; /* Fixed formatted text:
kana                 */
    unsigned char    voice[MSL_VOICE_SIZE] ;    /* Recognition
word                 */
} _MSL_TEIDATA_KOTEI ;
```

11.14 Deleting Free Formatted Texts (Range Specification)

Classification	General function		
Function name	Deleting free formatted texts (range specification)	Symbol	Msl_OpeUserSentenceDeleteAll
Function overview	<p>This function is used to delete a specified free formatted text. (Range: Formatted text numbers 21 to 50).</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeUserSentenceDeleteAll(s_no, e_no);		
Argument name	Type	I/O	Explanation
s_no	unsigned char	I	Number to be initialized (21 to 50)
e_no	unsigned char	I	
Return value	Type	I/O	Explanation
ret	Int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks	<p>Currently, only the following range is supported. If other than below is specified, an abnormal end occurs.</p> <p>Formatted text numbers 21 to 30 (folder 3) 31 to 40 (folder 4) 41 to 50 (folder 5)</p> <p>To delete (initialize) formatted text numbers 1 to 20, use Msl_OpeFreeUserSentenceSetRef(Registering or Referencing Free Formatted Texts) to write fixed values.</p>		

11.15 Registering or Referencing Free Formatted Texts

Classification	General function		
Function name	Registering or referencing free formatted texts	Symbol	Msl_OpeFreeUserSentenceSetRef
Function overview	<p>This function is used to read a free formatted text of the specified formatted text number.</p> <p>Formatted text numbers</p> <p>1 to 20: Fixed formatted text (20 types, search word, including kana formatted text):</p> <p>Out of target</p> <p>21 to 50: Free formatted text (30 types, no search word)</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeFreeUserSentenceSetRef(mode, no, data) ;		
Argument name	Type	I/O	Explanation
mode	Int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
no	unsigned char	I	Free formatted text number (21 to 50)
data	_MSL_TEIDATA_FREE *	O	Free formatted text structure pointer
Return value	Type	I/O	Explanation
ret	Int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks	<p>The formatted texts for folders 1 to 5 are as follows:</p> <p>Folder 1: Formatted text numbers 1 to 10</p> <p>Folder 2: Formatted text numbers 11 to 20</p> <p>Folder 3: Formatted text numbers 21 to 30</p> <p>Folder 4: Formatted text numbers 31 to 40</p> <p>Folder 5: Formatted text numbers 41 to 50</p> <p>[Free formatted text structure]</p> <pre>typedef struct tagMSL_TEIDATA_FREE { unsigned char entry_flg ; /* Flag indicating with or without user</pre>		

```
registration      */
    unsigned char  tei_kan[MSL_TEIKEI_MAX+1] ;    /* Free formatted text:
kanji              */
} _MSL_TEIDATA_FREE ;
```

11.16 Setting or Referencing Compact Desktop Information 2

Classification	General function		
Function name	Setting or referencing compact desktop information 2	Symbol	Msl_OpeLIMIconSetRef2
Function overview	<p>This function is used to set or reference object icon information.</p> <ul style="list-style-type: none"> - 15 objects are supported. 		
Include file	res_ope.h		
Calling sequence	int Msl_OpeLIMIconSetRef2(mode, dat) ;		
Argument name	Type	I/O	Explanation
mode	Int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
dat	_MSL_OBJICON2 *	I/O	Object information structure pointer (*1)
Return value	Type	I/O	Explanation
ret	Int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks	<p>*1 [Object information structure] typedef struct tagMSL_OBJICON2 { int Dummy; /* Dummy */ unsigned char ObjIcon_Inf[MSL_OBJICONALL2][316]; /* Icon information */ unsigned char Last_one_Icon; /* Last one icon position information */ } _MSL_OBJICON2; #define MSL_OBJICONALL2 15</p> <p>1) Data that can be specified in the ObjIcon_Inf member is as follows:</p>		


```

<Telephone number>      [15 members can be registered.]
    Icon type            (1 byte per member)
    Telephone number     (26 bytes per member)
    Name                 (32 bytes per member)
<URL>                   [15 members can be registered.]
    Icon type            (1 byte per member)
    URL address          (256 bytes per member)
    Title                (32 bytes per member)
<Address>               [15 members can be registered.]
    Icon type            (1 byte per member)
    Mail address         (50 bytes per member)
    Title                (1 title = 32 bytes)
<i-Appli>               [15 members can be registered.]
    Icon type            (1 byte per member)
    Unique ID            (4 bytes per member)
    i-Appli title        (32 bytes per member)
    ADF file URL         (255 bytes per member)
<Image>                 [15 members can be registered.]
    Icon type            (1 byte per member)
    Data number          (2 bytes per member)
    Image title          (32 bytes per member)
    With or without NVA sound (1 byte per member)
    Reference enabled or disabled (2 bytes per member)
<Melody>                [15 members can be registered.]
    Icon type            (1 byte per member)
    Tone type            (2 bytes per member)
    Melody title         (32 bytes per member)
    Reference enabled or disabled (2 bytes per member)
<Original menu>         [1 member can be registered.]
    Icon type            (1 byte per member)
    Title                (32 bytes per member)
    Reference enabled or disabled "2 bytes per member"
<ToDo>                  [1 member can be registered.]
    Icon type            (1 byte per member)
    Title                ( 32 bytes per member)
<video>                 [15 members can be registered.]
    Icon type            (1 byte per member)
    Data number          (2 bytes per member)
    Title                (32 bytes per member)
    Flag indicating with or without sound (1 byte per member)
    Camera or download type (1 byte per member)
    Reference enabled or disabled (2 bytes per member)
<Camera>                [1 member can be registered.]
    Icon type            (1 byte per member)
    Title                (32 bytes per member)

```

<Character recognition (OCR)> [1 member can be registered.]

Icon type (1 byte per member)

Title (32 bytes per member)

<Barcode> [1 member can be registered.]

Icon type (1 byte per member)

Title (32 bytes per member)

Last one icon position information (1 byte)

*** Specify a null at the end of the ObjIcon Inf member.**

*** Always register @ simultaneously with the URL or mail address.**

- The following values can be specified for the icon types of each object:

```
#define MSL_OBJICON_NO /* No data */
#define MSL_OBJICON_TELNO /* Telephone number */
#define MSL_OBJICON_URL /* URL */
#define MSL_OBJICON_MAIL /* Mail address */
#define MSL_OBJICON_MAILSMS /* SMS mail address */
#define MSL_OBJICON_IAPRI /* i-Appli */
#define MSL_OBJICON_MELODY /* Melody */
#define MSL_OBJICON_IMAGE /* Image */
#define MSL_OBJICON_MOVIE /* movie */
#define MSL_OBJICON_ORGMENU /* Original menu */
#define MSL_OBJICON_TODO /* TODO */
#define MSL_OBJICON_CAMERA /* Camera */
#define MSL_OBJICON_OCR /* Character recognition (OCR) */
#define MSL_OBJICON_BARCODE /* Barcode */
```

2) The last one icon position information that can be specified for Last_one_Icon is as follows:

```
#define MSL_LASTICON_DEFAULT /* Default position */
#define MSL_LASTICON_POS1 /* Position 1 */
#define MSL_LASTICON_POS2 /* Position 2 */
#define MSL_LASTICON_POS3 /* Position 3 */
#define MSL_LASTICON_POS4 /* Position 4 */
#define MSL_LASTICON_POS5 /* Position 5 */
#define MSL_LASTICON_POS6 /* Position 6 */
#define MSL_LASTICON_POS7 /* Position 7 */
#define MSL_LASTICON_POS8 /* Position 8 */
#define MSL_LASTICON_POS9 /* Position 9 */
#define MSL_LASTICON_POS10 /* Position 10 */
```

```
#define MSL_LASTICON_POS11    /* Position 11    */
#define MSL_LASTICON_POS12    /* Position 12    */
#define MSL_LASTICON_POS13    /* Position 13    */
#define MSL_LASTICON_POS14    /* Position 14    */
#define MSL_LASTICON_CALN     /* Calendar      */
```

11.17 Deleting Compact Desktop 2

Classification	General function		
Function name	Deleting compact desktop 2	Symbol	Msl_OpeLIMIconDel2
Function overview	<p>This function is used to delete an object icon. If there is data after the icon, justify the data.</p> <ul style="list-style-type: none"> - 15 objects are supported. 		
Include file	Res_ope.h		
Calling sequence	int Msl_OpeLIMIconDel2(no) ;		
Argument name	Type	I/O	Explanation
no	unsigned char	I	Deletion number
Return value	Type	I/O	Explanation
ret	Int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks	<p>Area to be set or referenced: NAND FLASH</p> <p>Deletion numbers:</p> <pre> #define MSL_OBJICON1 /* Deleting No.1 data */ #define MSL_OBJICON2 /* Deleting No.2 data */ #define MSL_OBJICON3 /* Deleting No.3 data */ #define MSL_OBJICON4 /* Deleting No.4 data */ #define MSL_OBJICON5 /* Deleting No.5 data */ #define MSL_OBJICON6 /* Deleting No.6 data */ #define MSL_OBJICON7 /* Deleting No.7 data */ #define MSL_OBJICON8 /* Deleting No.8 data */ #define MSL_OBJICON9 /* Deleting No.9 data */ #define MSL_OBJICON10 /* Deleting No.10 data */ #define MSL_OBJICON11 /* Deleting No.11 data */ #define MSL_OBJICON12 /* Deleting No.12 data */ #define MSL_OBJICON13 /* Deleting No.13 data */ #define MSL_OBJICON14 /* Deleting No.14 data */ #define MSL_OBJICON15 /* Deleting No.15 data */ #define MSL_OBJICONALL2 /* Deleting all data items */ </pre>		

11.18 Setting or Referencing Image Position

Classification	General function		
Function name	Setting or referencing image position	Symbol	Msl_OpeOrgImgPosSetRef
Function overview	<p>When the image size is larger than the image display size on the paste destination screen, this function allows the user to select a position to which the image is pasted. Select one from display from above, display center, and display from beneath. This function is used set or reference the display position that can be specified for each image.</p> <p>This function is also used to set or reference the image cut position (top, middle, or bottom) to register an original image.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeOrgImgPosSetRef(mode, func_id, num, data) ;		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
func_id	int	I	Function number: (*1) MSL_ORG_IMG_DSP_POS: Image display position MSL_ORG_IMG_CUT_POS: Image cut position
num	unsigned short	I	Image number
data	unsigned char *	I/O	Sets a display position. MSL_POS_UPPER: Display from above MSL_POS_CENTER: Display center MSL_POS_LOWER: Display from beneath
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			

When no image is registered in the original image number specified in num, this function writes a setting value to the SRAM. The upper-level application must determine whether the image is registered.

*1 The image cut position and image display position information correspond to 400 images respectively.

11.19 Setting or Referencing APN Data 2

Classification	General function		
Function name	Setting or referencing APN data 2	Symbol	Msl_OpeAPNSetRef2
Function overview	<p>This function is used to set or reference APN data.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeAPNSetRef2(mode, data) ;		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
Data	_MSL_OPEAPN2 *	I/O	APN data structure pointer Connection destination number: 1 byte MSL_DEFAULT_APN: Default setting (within the mobile phone) MSL_DEFAULT_APN_UIM: Default setting (within UIM) MSL_USR_APN1: User definition 1 MSL_USR_APN2: User definition 2 MSL_USR_APN3: User definition 3 MSL_USR_APN4: User definition 4 MSL_USR_APN5: User definition 5 MSL_USR_APN6: User definition 6 MSL_USR_APN7: User definition 7 MSL_USR_APN8: User definition 8 MSL_USR_APN9: User definition 9 MSL_USR_APN10: User definition 10 APN name: 99 byte APN name size: 1 byte Host name: 30 bytes Host name size: 1 byte Title: 18 bytes Title name size: 1 byte DNS name: 76 bytes DNS size: 1 byte
Return value	Type	I/O	Explanation
Ret	int	O	Processing result MSL_DEFAPN: Default mode being set (for reference)

			MSL_USRAPN: User mode being set (for reference)
			MSL_OK: Normal end
			MSL_NG: Abnormal end
Remarks			
<pre>[APN data structure] typedef struct _tag_MSL_OPEAPN2 { unsigned char mode; /* Connection destination number */ /* MSL_DEFAULT_APN: Default setting (within the mobile phone) */ /* MSL_DEFAULT_APN_UIM: Default setting (within UIM) */ /* MSL_USR_APN1: User definition 1 */ /* MSL_USR_APN2: User definition 2 */ /* MSL_USR_APN3: User definition 3 */ /* MSL_USR_APN4: User definition 4 */ /* MSL_USR_APN5: User definition 5 */ /* MSL_USR_APN6: User definition 6 */ /* MSL_USR_APN7: User definition 7 */ /* MSL_USR_APN8: User definition 8 */ /* MSL_USR_APN9: User definition 9 */ /* MSL_USR_APN10: User definition 10 */ unsigned char apn[MSL_APN_MAX]; /* APN name (connection destination address) */ unsigned char apn_len; /* APN name size */ unsigned char host[MSL_HOST_MAX]; /* Host name (connection destination name) */ unsigned char host_len; /* Host name size */ unsigned char apnname[MSL_APNNAME_MAX]; /* Title */ unsigned char apnname_len; /* Title name size */ unsigned char dnsname[MSL_DNSNAME_MAX]; /* DNS name */ unsigned char dnsname_len; /* DNS name size */ unsigned char dummy; /* Dummy */ } _MSL_OPEAPN2; #define MSL_APN_MAX 99 /* Upper limit of APN name data size */ #define MSL_HOST_MAX 30 /* Upper limit of host name data size */ #define MSL_APNNAME_MAX 18 /* Maximum APN name size */ #define MSL_DNSNAME_MAX 76 /* Maximum DNS name size */</pre>			

11.20 Setting or Referencing Color Name or RGB Information

Classification	General function		
Function name	Setting or referencing color name or RGB information	Symbol	Msl_OpeRgbSetRef
Function overview	<p>This function is used to set or reference the color name or RGB adjustment value of the specified color code.</p> <p>Note: If no data is set for the color name, set a null.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeRgbSetRef(mode, color, data);		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
color	unsigned char	I	Color code MSL_LEDCOLOR1: Color 1 MSL_LEDCOLOR2: Color 2 MSL_LEDCOLOR3: Color 3 : : MSL_LEDCOLOR11: Color 11 MSL_LEDCOLOR12: Color 12
data	_MSL_RGB_INF *	I/O	RGB setting or reference data storage area (*1)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (out of range of specified modes or color codes)
Remarks			

```
*1
[RGB setting or reference data structure]
typedef struct tagMSL_RGB_INF
{
    unsigned char    name [ MSL_RGB_NAME_LEN + 1 ];    /* Color
name */
    unsigned char    r;                                /* Luminance of red */
    unsigned char    g;                                /* Luminance of green */
    unsigned char    b;                                /* Luminance of blue */
} _MSL_RGB_INF ;

#define MSL_RGB_NAME_LEN 20 /* Color name
size */
```

11.21 Setting or Referencing Order of Original Animations 2

Classification	General function		
Function name	Setting or referencing order of original animations 2	Symbol	Msl_OpeOrglmgAnimeOrdSetRef2
Function overview	<p>This function is used to set or reference the display position of the user-defined original animation (20 positions).</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeOrglmgAnimeOrdSetRef2(mode, no) ;		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
no	int	I	Self-produced animation number MSL_OPE_OWNaNIME01 Self-produced animation 1 ~ MSL_OPE_OWNaNIME20 Self-produced animation 20
order	unsigned short *	I/O	Animation display order storage area
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (out of range of specified modes)
Remarks			

11.22 Setting or Referencing Melody Player Program List

Classification	General function		
Function name	Setting or referencing melody player program list	Symbol	Msl_OpeMelodyPlayListSetRef
Function overview	<p>This function is used to save or reference the program list created with melody player program editing.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeMelodyPlayListSetRef (mode, list_num, data) ;		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
list_num	unsigned char	I	List data number, 10 numbers (0 to 9)
data	_MSL_OPEMELOPLAYER *	I/O	Melody player data structure pointer (*1)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (out of range of specified modes)
Remarks	<p>*1 [Melody player data structure] typedef struct tagMSL_OPEMELOPLAYER{ int mode; /* mode: Sound type */ short SoundID; /* SoundID: Sound ID */ }_MSL_OPEMELOPLAYER ;</p>		

11.23 Deleting Melody Player Program List

Classification	General function		
Function name	Deleting melody player program list	Symbol	Msl_OpeMelodyPlayListDel
Function overview	<p>This function is used to delete a program list of the specified number.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeMelodyPlayListDel (listnum)		
Argument name	Type	I/O	Explanation
Listnum	unsigned char	I	Program list number, 10 numbers (0 to 9)
Return value	Type	I/O	Explanation
Ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (out of range of specified modes)
Remarks			

11.24 Deleting All Items on Melody Player Program List

Classification	General function		
Function name	Deleting all items on melody player program list	Symbol	Msl_OpeMelodyPlayListDelAll
Function overview	<p>This function is used to delete all items on the program list.</p>		
Include file	res_ope.h		
Calling sequence	void Msl_OpeMelodyPlayListDelAll(void)		
Argument name	Type	I/O	Explanation
-	void	-	-
Return value	Type	I/O	Explanation
-	void	-	-
Remarks			

11.25 Operating (Setting, Referencing, or Deleting) Mailing List Name

Classification	General function		
Function name	Operating (setting, referencing, or deleting) mailing list name	Symbol	Msl_Ope_ML_NameCtl
Function overview	<p>This function is used to set, reference, or delete the name of the specified mailing list.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_Ope_ML_NameCtl(mode, list_no, data)		
Argument name	Type	I/O	Explanation
mode	int	I	Operation mode MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing MSL_FUNCDEL: Deletion processing
list_no	int	I	Mailing list number MSL_OPE_ML_LISTNO_00: List number 0 ~ MSL_OPE_ML_LISTNO_19: List number 19
data	unsigned char *	I/O	Mailing list name storage area 21 bytes (including a null at the end)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks	<p>Notes:</p> <ul style="list-style-type: none"> - The mailing list name must consist of up to 20 characters (21 bytes including a null at the end). The upper-level application must set a null at the end of the mailing list. - The mailing list number is provided using a define. 		

- When the specified data consists of more than 18 bytes, the system internally specifies a null in byte 19 before storing the data.
When data uses a 2-byte character, the system does not guarantee the data.
- When null data is specified at the beginning of the setting data, this function assumes that the mailing list name is to be deleted.
- When no mailing list name is specified, the system sets a null character string in the storage area.

Details of the processing result:

MSL_OK: Normal end

When the stored mailing list name can be set, referenced, or deleted correctly:

MSL_NG: Abnormal end

- When the specified mailing list number is out of range:
MSL_OPE_ML_LISTNO_00 to MSL_OPE_ML_LISTNO_19
(0 to 19)
- When a NAND nonvolatile write results in an error

11.26 Operating (Setting, Referencing, or Deleting) Mail Address

Classification	General function		
Function name	Operating mail address	Symbol	Msl_Ope_ML_AddrSetRef
Function overview	<p>This function is used to set, reference, or delete a specified mail address on the specified mailing list.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_Ope_ML_AddrSetRef(mode, list_no, addr_no, addr_buf)		
Argument name	Type	I/O	Explanation
mode	int	I	Operation mode MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing MSL_FUNCDEL: Deletion processing
list_no	int	I	Mailing list number MSL_OPE_ML_LISTNO_00: List number 0 ~ MSL_OPE_ML_LISTNO_19: List number 19
addr_no	int	I	Mail address number MSL_OPE_ML_ADDNO_0: Address number 0 ~ MSL_OPE_ML_ADDNO_4: Address number 4
addr_buf	unsigned char *	I/O	Mail address storage area of 51 bytes (including a null at the end)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			

Notes:

- The mail address must consist of up to 50 characters (51 bytes including a null at the end).
The upper-level application must set a null at the end of the mail address.
- When the specified data consists of more than 50 bytes, the system internally specifies a null in byte 51 before storing the data.
- When null data is specified at the beginning of the setting data, this function assumes that the mail address is to be deleted.
- The mailing list number and mail address are provided using defines.
- When no mail address is specified, the system sets a null character string in the storage area.

Details of the processing result:

MSL_OK: Normal end:

When the stored mail address can be set, referenced, or initialized normally

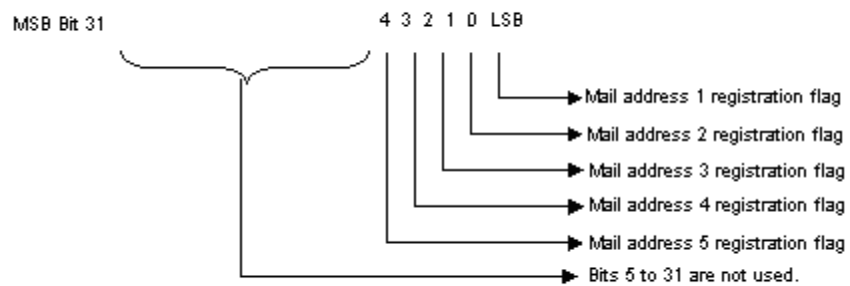
MSL_NG: Abnormal end:

- When the specified mailing list number is out of range
MSL_OPE_ML_LISTNO_00 to MSL_OPE_ML_LISTNO_19
(0 to 19)
- When the specified mail address number is out of range
MSL_OPE_ML_ADDNO_0 to MSL_OPE_ML_ADDNO_4
(0 to 4)
- When a NAND nonvolatile write or read results in an error

11.27 Referencing Whether Mail Address Is Registered

Classification	General function		
Function name	Referencing whether mail address is registered	Symbol	Msl_Ope_ML_AddrFlgRef
Function overview	<p>This function is used to reference the mail address registration flag on the specified mailing list.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_Ope_ML_AddrFlgRef(list_no, addrflg)		
Argument name	Type	I/O	Explanation
list_no	Int	I	<p>Mailing list number</p> <p>MSL_OPE_ML_LISTNO_00: List number 0</p> <p>~</p> <p>MSL_OPE_ML_LISTNO_19: List number 19</p>
addrflg	int *	O	Mail address registration flag storage area
Return value	Type	I/O	Explanation
Ret	Int	O	<p>Processing result</p> <p>MSL_OK: Normal end</p> <p>MSL_NG: Abnormal end</p>
Remarks			

Details of mail address registration flag



* Bit on indicates that a mail address is registered.

Notes:

- The mailing list number id provided using a define.

Details of the processing result:

MSL_OK: Normal end

When the registration flag of the stored mailing list can be collected normally

MSL_NG: Abnormal end

When the specified mailing list number is out of range

MSL_OPE_ML_LISTNO_00 to MSL_OPE_ML_LISTNO_19
(0 to 19)

11.28 Referencing (Listing) Mail Address

Classification	General function		
Function name	Referencing (listing) mail address	Symbol	Msl_Ope_ML_AddrListRef
Function overview	<p>This function is used to reference the name of the specified mailing list.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_Ope_ML_AddrListRef(list_no, addr_buf)		
Argument name	Type	I/O	Explanation
list_no	int	I	<p>Mailing list number</p> <p>MSL_OPE_ML_LISTNO_00: List number 0</p> <p>~</p> <p>MSL_OPE_ML_LISTNO_19: List number 19</p>
addr_buf	_MSL_OPE_ML_ADRLST *	O	Pointer of the mail address reference list structure
Return value	Type	I/O	Explanation
ret	int	O	<p>Processing result</p> <p>MSL_OK: Normal end</p> <p>MSL_NG: Abnormal end</p>
Remarks	<p>Notes:</p> <ul style="list-style-type: none"> - The mail address must consist of up to 50 characters (51 bytes including a null at the end). - The mailing list number is provided using a define. - Slip out data is output without being changed. - When an NAND nonvolatile read error occurs, specify a null. <p>Mail address storage structure:</p>		

The following shows the details of the mail address storage structure:

[Mail address reference list structure]

```
typedef struct tagMSL_OPE_ML_ADRLST
{
    int      addr_cnt ;           /* Number of mail addresses
registered */
    int      addr_flg ;           /* Registration flag */
    unsigned
char    mail_addr[MSL_OPE_ML_ADDR_MAX][MSL_OPE_ML_ADDRNAME_MAX
+ 1] ;
                                /* Address for one list */
} _MSL_OPE_ML_ADRLST ;

#define MSL_OPE_ML_ADDR_MAX    5           /* Maximum number of mail
addresses on the mailing list */
#define MSL_OPE_ML_ADDRNAME_MAX  50       /* Maximum mail
address size */
```

Details of the processing result:

```
MSL_OK:  Normal end
          When the stored mail address can be referenced correctly

MSL_NG:  Abnormal end
          When the specified mailing list number is out of range
          MSL_OPE_ML_LISTNO_00 to MSL_OPE_ML_LISTNO_19
          (0 to 19)
```

11.29 Referencing Number of Mail Addresses Registered

Classification	General function		
Function name	Referencing number of mail addresses registered	Symbol	Msl_Ope_ML_AddrCntRef
Function overview	<p>This function is used to reference the number of mail addresses registered on the specified mailing list.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_Ope_ML_AddrCntRef(list_no, cnt)		
Argument name	Type	I/O	Explanation
list_no	Int	I	<p>Mailing list number</p> <p>MSL_OPE_ML_LISTNO_00: List number 0</p> <p>~</p> <p>MSL_OPE_ML_LISTNO_19: List number 19</p>
cnt	int *	O	Number of mail addresses registered
Return value	Type	I/O	Explanation
ret	Int	O	<p>Processing result</p> <p>MSL_OK: Normal end</p> <p>MSL_NG: Abnormal end</p>
Remarks	<p>Notes:</p> <ul style="list-style-type: none"> - The mailing list number is provided using a define. <p>Details of processing result:</p> <p>MSL_OK: Normal end</p> <p>When the specified mail address can be initialized normally</p> <p>MSL_NG: Abnormal end</p> <p>When the specified mailing list number is out of range</p>		

MSL_OPE_ML_LISTNO_00 to MSL_OPE_ML_LISTNO_19
(0 to 19)

11.30 Referencing Total Number of Mail Addresses Registered

Classification	General function		
Function name	Referencing total number of mail addresses registered	Symbol	Msl_Ope_ML_AddrCntAllRef
Function overview	<p>This function is used to reference the total number of mail addresses registered on the mailing list.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_Ope_ML_AddrCntAllRef(void)		
Argument name	Type	I/O	Explanation
-	Void	-	-
Return value	Type	I/O	Explanation
ret	Int	O	Total number of mail addresses registered 0 to 100
Remarks			

11.31 Setting or Referencing Alarm Suspension Information 2

Classification	Clock function setting		
Function name	Setting or referencing alarm suspension information 2	Symbol	Msl_OpeAlmHoldSetRef2
Function overview	<p>This function is used to set or reference alarm suspension information.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeAlmHoldSetRef2(mode, data)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
data	_MSL_OPE_ALMHOLD2 *	I/O	Alarm suspension information pointer
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks	<p>Notes: Specify a null in the initial value of alarm suspension information structure 2.</p> <p>Details of alarm suspension information structure 2 are as follows:</p> <p>[Alarm suspension information structure] typedef struct tagMSL_OPE_ALMHOLD2 { unsigned short alm_hold ; /* Alarm suspension type flag */ unsigned short dummy ; /* Supporting 2-byte boundary */ char alm_hold_date1[MSL_OPE_ALMHOLD_DATELEN] ; /* Alarm suspension date and time 1 (schedule) */ char alm_hold_date2[MSL_OPE_ALMHOLD_DATELEN] ; /* Alarm suspension date and time 2 (wakeup) */ char alm_hold_date3[MSL_OPE_ALMHOLD_DATELEN] ; /* Alarm suspension date and time 3 (ToDo) */ char alm_hold_msg[MSL_OPE_ALMHOLD_MSGLEN2];/* Alarm suspension message content (schedule) */</p>		

```

    char      alm_hold_msg_todo[MSL_OPE_ALMHOLD_MSGLEN2];/* Alarm
suspension message content (ToDo)*/
    int      alm_hold_day1 ;          /* Alarm suspension day of the week 1
(schedule) */
    int      alm_hold_day2 ;          /* Alarm suspension day of the week 2
(wakeup) */
    int      alm_hold_day3 ;          /* Alarm suspension day of the week 3
(ToDo) */
    unsigned char alm_hold_pri ; /* Alarm suspension priority (ToDo) */
    unsigned cahr alm_hold_sts ; /* Alarm suspension state (ToDo) */
} _MSL_OPE_ALMHOLD2 ;

```

```

#define MSL_OPE_ALMHOLD_DATELEN  24 /* Alarm suspension date and
time size */

```

```

#define MSL_OPE_ALMHOLD_MSGLEN2  100 /* Alarm suspension message
size */

```

Details of the processing result:

MSL_OK: Normal end

When alarm suspension information saved to the nonvolatile memory can be set or referenced normally

MSL_NG: Abnormal end

When the operation specification is out of range or alarm suspension information pointer is a null

11.32 Setting or Referencing SMScenter (User-defined)

Classification	Communication function setting		
Function name	Setting or referencing SMScenter (user-defined)	Symbol	Msl_OpeSMScenterSetRef
Function overview			
This function is used to set or references user-defined information of the SMScenter setting functions.			
Include file	res_ope.h		
Calling sequence	int Msl_OpeSMScenterSetRef(mode, usrdata)		
Argument name	Type	I/O	Explanation
mode	Int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
usrdata	unsigned char *	I/O	User setting information (*1)
Return value	Type	I/O	Explanation
ret	Int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			
Notes:			
<ul style="list-style-type: none">- The maximum number of characters that can be entered for the user-defined information must be 20 bytes (20 digits).- Use one byte to specify the SMS center.- Data validity is not checked.			
*1 Details of 22-byte data in the user-defined information are as shown below.			
<div><div><div>:usrdata[0]~usrdata[19]</div><div><div>SMS center address</div><div>NULL</div><div>SMS center specification (International or Unknown)</div></div></div><div><div>:usrdata[20]</div><div>:usrdata[21]</div></div></div> <ul style="list-style-type: none">- The initial value of the user-defined information must be a null.			
Details of the processing result			

-MSL_OK: Normal end

When the user definition can be set or referenced normally

-MSL_NG: Abnormal end

When operation specification mode is other than MSL_FUNCSET or
MSL_FUNCREF

11.33 Setting or Referencing Certificate Download

Classification	Terminal function		
Function name	Setting or referencing certificate download	Symbol	Msl_OpeCertificateDLSetRef
Function overview	<p>This function is used to set or reference certificate download data.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeCertificateDLSetRef(mode, data)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
data	_MSL_OPE_CERTIFICATEAPN *	I/O	Certificate download information structure pointer Connection number: 1 byte (0: Default setting, 1: User data setting) APN name: 99 bytes APN name size: 1 byte Host name: 100 bytes Host name size: 1 byte
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_DEFAPN_CERTIFICATE : Default mode is being set (for reference) MSL_USRAPN_CERTIFICATE : User mode is being set (for reference) MSL_OK: Normal end (for setting) MSL_NG: Abnormal end (parameter NG)(*1) *1 When data -> mode is other than 0 or 1 (for setting or reference)

Remarks	
	<pre> [Certificate download information structure] typedef struct _tag_MSL_OPE_CERTIFICATEAPN { unsigned char mode; /* Connection destination number */ /* 0: Default setting */ /* 1: User setting */ unsigned char apn[MSL_APN_MAX]; /* APN name (connection destination address) */ unsigned char apn_len; /* APN name size */ unsigned char host[MSL_CERTIFIHOST_MAX]; /* Host name (connection destination name) */ unsigned char host_len; /* Host name size */ } _MSL_OPE_CERTIFICATEAPN; #define MSL_APN_MAX 99; /* Upper limit of the APN name data size */ #define MSL_CERTIFIHOST_MAX 100; /* Maximum host name size */ </pre>

11.34 Setting or Referencing Various Ring Tone Patterns for Using TV Phone

Classification	Terminal function		
Function name	Setting or referencing various ring tone patterns for using TV phone	Symbol	Msl_OpeTVTel_RcvToneSetRef
Function overview	<p>This function is used to set or reference the ring tone patterns for using TV phone.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeTVTel_RcvToneSetRef(mode, soundid)		
Argument name	Type	I/O	Explanation
Mode	Int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
soundid	int *	I/O	Sound ID
Return value	Type	I/O	Explanation
Ret	Int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			

11.35 Setting or Referencing Communication Mode

Classification	Terminal function		
Function name	Setting or referencing communication mode	Symbol	Msl_OpeTVTel_ComModeSetRef
Function overview	<p>This function is used to set or reference communication mode for using the TV phone.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeTVTel_ComModeSetRef(mode, commode)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
commode	int *	I/O	Communication mode (*1) MSL_TVTELCOMMODE_MOVE: Motion priority MSL_TVTELCOMMODE_STD: Standard MSL_TVTELCOMMODE_QUALITY: Image quality priority
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks	<p>*1 Set communication mode for using the TV phone.</p>		

11.36 Setting or Referencing Various TV Phone Information Items

Classification	Terminal function		
Function name	Setting or referencing various TV phone information items	Symbol	Msl_OpeTVTel_StatusSetRef
Function overview	<p>This function is used to set or reference the following information for using the TV phone:</p> <ul style="list-style-type: none"> - Setting or referencing outgoing self-image transmission - Setting or referencing an alternative image - Setting or referencing a fallback - Response suspension image - Communication suspension image - Alternative image for animation memo - TV message slip image - 64k or 32k outgoing selection - Voice re-calling - Parent-child screen switching status - TV phone incoming rejection 		
Include file	res_ope.h		
Calling sequence	int Msl_OpeTVTel_StatusSetRef(mode, func_id, status)		
Argument name	Type	I/O	Explanation
mode	Int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
func_id	Int	I	Selects a function. MSL_OPE_TV_TEL_SND_SELFIMG: Outgoing self-image transmission MSL_OPE_TV_TEL_SUBIMG: Alternative image MSL_OPE_TV_TEL_FALLBACK: Fallback MSL_OPE_TV_RPLY_SPNDIMG: Response suspension image MSL_OPE_TV_COM_SPNDIMG: Communication suspension image MSL_OPE_TV_MOVIE_SUBIMG: Alternative image for animation memo MSL_OPE_TV_TVMEMOIMG: TV message slip image MSL_OPE_TV_64K32K: 64k or 32k

			outgoing selection MSL_OPE_TV_VOICE_RESND: Voice re-calling MSL_OPE_TV_CHANGEWINDOW: Parent-child screen switching status MSL_OPE_TV_REFUSAL: TV phone incoming rejection
status	int *	I/O	Selects an image. (*1) - Outgoing self-image transmission, voice re-calling, and TV phone incoming rejection: MSL_ON: Set MSL_OFF: Cancelled - Alternative image, response suspension image, communication suspension image, alternative image for animation memo, TV message slip image: MSL_TVTEL_USERIMG: Self-produced (user) MSL_TVTEL_HOLDIMG: Built-in - Fallback MSL_ON: Set MSL_OFF: Cancelled - 64k or 32k outgoing selection (outgoing type) MSL_TVTEL_64K: 64K MSL_TVTEL_32K: 32K - Parent-child screen switching status: MSL_TVTEL_WND1: Mode 1 MSL_TVTEL_WND2: Mode 2 MSL_TVTEL_WND3: Mode 3 MSL_TVTEL_WND4: Mode 4 MSL_TVTEL_WND5: Mode 5 MSL_TVTEL_WND6: Mode 6 MSL_TVTEL_WND7: Mode 7
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end

			MSL_NG: Abnormal end
Remarks			
<p>*1 This value is reflected only for setting.</p>			

11.37 Setting or Referencing Remote Monitoring

Classification	Terminal function		
Function name	Setting or referencing remote monitoring	Symbol	Msl_OpeTVTel_RemoteSetRef
Function overview	<p>This function is used to set or reference remote monitoring for using the TV phone.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeTVTel_RemoteSetRef(mode, msl_remote_monitor)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
msl_remote_monitor	_MSL_REMOTE_MONITOR *	I/O	Remote monitoring setting structure pointer
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks	<pre>[Remote monitoring setting structure] typedef struct tagMSL_REMOTE_MONITOR { int RemoteStatus; /* Remote monitoring */ int RemoteTime; /* Monitoring time */ unsigned char Dial01[MSL_TVTEL_DIAL_LEN]; /* Registration number 1 */ unsigned char Dial02[MSL_TVTEL_DIAL_LEN]; /* Registration number 2 */ unsigned char Dial03[MSL_TVTEL_DIAL_LEN]; /* Registration number 3 */ unsigned char Dial04[MSL_TVTEL_DIAL_LEN]; /* Registration number 4 */ unsigned char Dial05[MSL_TVTEL_DIAL_LEN]; /* Registration number 5 */ } _MSL_REMOTE_MONITOR; #define MSL_TVTEL_DIAL_LEN 28 /* Dial length */</pre>		

11.38 Setting or Referencing Continuous Image Titles

Classification	General function		
Function name	Setting or referencing continuous image titles	Symbol	Msl_OpeConPhotoTitleSetRef
Function overview	<p>This function is used to set or reference the continuous image titles.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeConPhotoTitleSetRef(mode , no , title);		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
no	unsigned char	I	Specification number (0 to 9)
title	unsigned char *	I/O	Continuous image title storage area Data size: 19 bytes (including a null at the end)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (out of range of specified modes or data specification numbers)
Remarks	<p>Title data size: unsigned char title[18+1];</p> <pre>#define MSL_CONPHOTOMAX 9 /* Maximum specification number */</pre>		

11.39 Setting or Referencing Order of Continuous Images

Classification	General function		
Function name	Setting or referencing order of continuous images	Symbol	Msl_OpeConPhotoOrderSetRef
Function overview	<p>This function is used to set or reference the display position of the continuous images.</p> <p>Up to 10 continuous images can be set.</p> <p>Up to 10 frames can be registered for a continuous image.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeConPhotoOrderSetRef(mode , no , order);		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
no	unsigned char	I	Specification number (0 to 9)
order	unsigned short *	I/O	Image order data storage area
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (out of range of specified modes or data specification numbers)
Remarks	<pre>#define MSL_CONPHOTOMAX 9 /* Maximum specification number */</pre> <p>- Specify a null for the frame position where no data is registered.</p>		

11.40 Deleting Continuous Images

Classification	General function		
Function name	Deleting continuous images	Symbol	Msl_OpeConPhotoDelete
Function overview	<p>Specify 0xffff in the area for the specified number. To specify all numbers, specify 0xffff in all areas of the 10 continuous images.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeConPhotoDelete(no) ;		
Argument name	Type	I/O	Explanation
No	unsigned char	I	Specification number (0 to 9) MSL_CONPHOTOALL: All numbers specified
Return value	Type	I/O	Explanation
Ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (out of range of specified numbers)
Remarks	<p>* Call this function in the following state:</p> <ul style="list-style-type: none"> - When canceling continuous images - When all frames are cancelled from images at overwrite editing - When the D1 command is used for initialization 		

11.41 Setting or Referencing Recording Image Quality

Classification	General function		
Function name	Setting or referencing recording image quality	Symbol	Msl_OpeCam_ReclmgQualitySetRef
Function overview	<p>This function is used to set or reference the recording image quality.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeCam_ReclmgQualitySetRef (mode , recmode, quality) ;		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
recmode	int	I	Recording mode MSL_CAM_RECSIZE_SNAP: Snap MSL_CAM_RECSIZE_MOVE: Recording mainframe mode MSL_CAM_RECSIZE_MOVE_SD: Recording SD mode
quality	int *	I/O	Image quality setting data MSL_CAM_LONG: Long-time mode MSL_CAM_NORMAL: Standard mode MSL_CAM_HIGH: High quality mode MSL_CAM_SUPER: Superfine mode
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (out of specification of specified modes or data specification numbers)
Remarks	<p>*1 Set camera recording quality data.</p>		

11.42 Setting or Referencing Recording Size Selection

Classification	General function		
Function name	Setting or referencing recording size selection	Symbol	Msl_OpeCam_RecSizeSetRef
Function overview	<p>This function is used to set or reference recording size selection.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeCam_RecSizeSetRef (mode , recmode , value)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
recmode	int	I	Recording mode MSL_CAM_REC_SIZE_SNAP: Snap MSL_CAM_REC_SIZE_MOVE: Recording mode
value	int *	I/O	Recording mode data (*1) MSL_CAM_REC_SIZE_MAIL: Mail-attached mode (100 Kbytes) MSL_CAM_REC_SIZE_FREE: Not limited MSL_CAM_REC_SIZE_LONG: Long-time mode
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks	<p>*1 Set camera recording mode data.</p>		

11.43 Setting or Referencing Continuous Shooting Function

Classification	General function		
Function name	Setting or referencing continuous shooting function	Symbol	Msl_OpeCam_RapTimeSetRef
Function overview	<p>This function is used to set or reference the continuous shooting function.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeCam_RapTimeSetRef (mode , value) ;		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
value	int *	I/O	Continuous shooting time data (*1) MSL_CAM_RAPTIME_SLOW: Slow (one second) MSL_CAM_RAPTIME_QUICK: Quick (0.4 seconds) MSL_CAM_RAPTIME_MATH: Mach (0.06 seconds)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks	<p>*1 Set continuous shooting time data.</p>		

11.44 Setting or Referencing Brightness Adjustment

Classification	General function		
Function name	Setting referencing brightness adjustment	or Symbol	Msl_OpeCam_BrightnessSetRef
Function overview	<p>This function is used to set or reference the brightness for using the TV phone.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeCam_BrightnessSetRef (mode , value) ;		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
value	int *	I/O	Brightness value (*1) MSL_CAM_BRIGHT_1: Adjustment value 1 MSL_CAM_BRIGHT_2: Adjustment value 2 MSL_CAM_BRIGHT_3: Adjustment value 3 MSL_CAM_BRIGHT_4: Adjustment value 4 MSL_CAM_BRIGHT_5: Adjustment value 5
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks	<p>*1 Specify the brightness of the TV phone.</p>		

11.45 Setting or Referencing UIM Information

Classification	General function		
Function name	Setting or referencing UIM information	Symbol	Msl_OpeUIMOpeSetRef
Function overview	<p>This function is used to set or reference UIM certificate valid or invalid information (EF_ACTIVATE).</p> <p>This function is used to set or reference UIM certificate cache information (EF_ROOT, FF_SUBROOT, or EF_CERT).</p> <p>This function is used to set or reference UIM data read result information.</p> <p>This function is used to set or reference UIM identification information (ICCID or IMODER) within the mobile phone.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeUIMOpeSetRef(mode , func_id , kind, data) ;		
Argument name	Type	I/O	Explanation
mode	Int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
func_id	Int	I	Selects a function. MSL_OPE_EFCTIVATE: UIM certificate valid or invalid information MSL_OPE_CASH: UIM certificate cache information MSL_OPE_READRES: UIM data read result information MSL_OPE_UIMIDINFO: UIM identification information within the mobile phone
kind	unsigned char	I	UIM certificate cache type: (*1) MSL_OPE_UIMCASH_EFROOT: EF_ROOT MSL_OPE_UIMCASH_EFSUBROOT: EF_SUBROOT MSL_OPE_UIMCASH_EFCERT: EF_CERT
data	unsigned char *	I/O	Pointer of the UIM certificate valid or invalid information storage (*2) Pointer of the UIM certificate cache information storage (*3)

			Pointer of the UIM data read result information storage (*4) UIM identification information storage pointer within the mobile phone (10 bytes)
Return value	Type	I/O	Explanation
ret	Int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (Read NG, Write NG, parameter NG)
Remarks	<p>*1 The UIM certificate cache type is reflected only when the UIM certificate cache information function is selected.</p> <p>*2 data[0]: User certificate valid or invalid data[1]: Route CA certificate valid or invalid</p> <p>Valid: MSL_OPE_UIMEFFECT(0x01)/Invalid: MSL_OPE_UIMNOEFFECT(0x02)</p> <p>*3 Data size => data[MSL_OPE_UIMCASH_MAXSIZE] MSL_OPE_UIMCASH_MAXSIZE(1500)</p> <p>*4 data[0] => Read result of EF_ACTIVATE (1 byte) data[1] => Read result of EF_ROOT (1 byte) data[2] => Read result of EF_SUBROOT (1 byte) data[3] => Read result of EF_CERT (1 byte)</p>		

11.46 Setting or Referencing EEPROM Area

Classification	Terminal setting section		
Function name	Setting or referencing EEPROM area	Symbol	Msl_OpeEEPROMSetRef
Function overview	<p>This function is used to set or reference EEPROM area data.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeEEPROMSetRef(mode , funcid , data) ;		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
funcid	unsigned short	I	Function ID of the data to be set or referenced
data	void *	I/O	Setting or reference data pointer
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (parameter error)
Remarks	<ul style="list-style-type: none"> - Only check mode (first argument) for the parameter check. * The calling source must allocate an area having the size that can store the data for the setting or reference data pointer. - For setting or referencing data, the calling source must have the type to be used. 		

11.47 Setting or Referencing Self-produced Animation Title

Classification	General function		
Function name	Setting or referencing self-produced animation title	Symbol	Msl_OpeOrgAnimeTitleSetRef
Function overview	<p>This function is used to set or reference a self-produced animation title.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeOrgAnimeTitleSetRef(mode , id , data) ;		
Argument name	Type	I/O	Explanation
Mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
Id	int	I	Image ID MSL_OPE_OWNaNIME01 Self-produced animation 1 ~ MSL_OPE_OWNaNIME20 Self-produced animation 20
Data	_MSL_OPE_ORGANIMEINFO *	I/O	Pointer of the self-produced animation title information structure
Return value	Type	I/O	Explanation
Ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (out of range of specified modes or image IDs)
Remarks	<pre>[Self-produced animation title information structure] typedef struct tagMSL_OPE_ORGANIMEINFO { unsigned char title[18+1]; /* Self-produced animation title storage area */ unsigned short infoflag; /* Additional information flag */ } _MSL_OPE_ORGANIMEINFO ;</pre>		

11.48 Setting Voice Clock

Classification	General function		
Function name	Setting voice clock	Symbol	Msl_OpeVoiceClockSetRef
Function overview	<p>This function is used to set or reference the voice clock.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeVoiceClockSetRef(mode , data) ;		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
data	_MSL_VOICE_CLOCK *	I/O	Voice clock data structure pointer (*1) On or off flag: 1 byte Voice or sound flag: 1 byte
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (out of range of specified modes)
Remarks	<pre> *1 [Voice clock data structure] typedef struct tagMSL_VOICE_CLOCK { unsigned char flg ; /* On or off flag */ unsigned char voiceflg ; /* Voice or sound flag */ } _MSL_VOICE_CLOCK ; /* Define of the voice or sound flag */ #define MSL_VOICE /* Voice */ #define MSL_SOUND /* Sound */ </pre>		

11.49 Setting or Referencing Animation Backlight

Classification	Terminal function		
Function name	Setting or referencing animation backlight	Symbol	Msl_OpeMovieBKLTSetRef
Function overview	<p>This function is used to set or reference the animation backlight.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeMovieBKLTSetRef (mode , bklt) ;		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
bklt	int *	I/O	Sets backlight (standard or always on) (*1) MSL_OPE_BKLT_STD: Standard MSL_OPE_BKLT_NOW: Always on
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (out of range of specified modes or image IDs)
Remarks	<p>*1 Set animation backlight data.</p>		

11.50 Setting or Referencing Communication Time

Classification	Communication function		
Function name	Setting or referencing communication time	Symbol	Msl_OpeComTalkTimeSetRef
Function overview	<p>This function provides the following functions:</p> <ul style="list-style-type: none"> - Sets or references the call type and communication time in the previous communication. - Sets or references the accumulated communication time of the specified call type. 		
Include file	res_ope.h		
Calling sequence	int Msl_OpeComTalkTimeSetRef (mode, func_id, tktime)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
func_id	int	I	Function type MSL_OPE_TALK_TIME: Operating previous communication time MSL_OPE_TALKTTL_TIME: Operating the accumulated communication time
tktime	_MSL_OPE_TKTIME *	I/O	Communication time data structure pointer (*1)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (parameter error)
Remarks	<p>The accumulated communication time (in seconds) that can be obtained using reference operation of the accumulated communication time must be 0 to 719,999 seconds.</p> <p>When the accumulated communication time exceeds 719,999, the difference between the time and 720,000 is set as accumulated communication time.</p>		

***1**

[Communication time data structure]

```
typedef struct tagMSL_OPE_TKTIME {
    int  kind ;      /* Call type (*2) */
                /* MSL_TK_DEGITAL:  Digital communication */
                /* MSL_TK_TV:   TV phone communication */
                /* MSL_TK_NONE:  No communication */
    long tk_time ;    /* Communication time (second) */
} _MSL_OPE_TKTIME ;
```

***2** When MSL_FUNCSET is specified in mode, IN information is assumed. When MSL_FUNCREF is specified, OUT information is assumed. However, when MSL_OPE_TALKTTL_TIME is specified in func_id, IN information is assumed even when MSL_FUNCREF is specified in mode.

```
/* Call type (kind) */
#define MSL_TK_DEGITAL /* Digital communication */
#define MSL_TK_TV /* TV phone communication */
```

11.51 Initializing Communication Time or Calling Rate

Classification	Communication function		
Function name	Initializing communication time or calling rate	Symbol	Msl_OpeSetInfRset
Function overview	<p>This function is used to reset (clear to 0) the communication time or calling rate data.</p> <ul style="list-style-type: none"> - Initializing communication time - Initializing accumulated communication time - Initializing calling rate - Initializing accumulated calling rates - Initializing all items 		
Include file	res_ope.h		
Calling sequence	int Msl_OpeSetInfRset (kind)		
Argument name	Type	I/O	Explanation
Kind	int	I	<p>Specifies an item to be initialized.</p> <p>MSL_TKTIME: Initializes communication time.</p> <p>MSL_TKTTLTIME: Initializes accumulated communication time.</p> <p>MSL_TKRATE: Initializes calling rate.</p> <p>MSL_TKTTLRATE: Initializes accumulated calling rates.</p> <p>MSL_TKALL: Initializes all items.</p>
Return value	Type	I/O	Explanation
Ret	int	O	<p>Processing result</p> <p>MSL_OK: Normal end</p> <p>MSL_NG: Abnormal end (parameter error)</p>
Remarks			

11.52 Setting or Referencing Voice or Message Slip Management Information

Classification	Recording function		
Function name	Setting or referencing voice or message slip management information	Symbol	Msl_OpeRecMsgInfSetRef
Function overview	<p>This function is used to set or reference voice or message slip management information.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeRecMsgInfSetRef(mode, data)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
data	_MSL_RECMSGINF *	I/O	Pointer of the voice or message slip management structure (*1)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (parameter error)
Remarks	<p>If an inconsistency occurs between the recorded voice data and management data to be set or referenced using this function, erase the voice data, update management information, and rewrite data.</p> <p>*1 [Voice or message slip management structure] typedef struct tagMSL_RECMSGINF { int play[MSL_REC_MAX]; /* Registration order of recorded data items */ int telmsg_free; /* Free message slip number (smallest) */ int next_rec; /* Next recording number of the voice slip */ } _MSL_RECMSGINF;</p>		

```
#define MSL_REC_MAX 6 /* Maximum number of recorded data  
items */
```


11.53 Registering or Referencing Service Setting Number

Classification	General function		
Function name	Registering or referencing service setting number	Symbol	Msl_OpeSrvInfoNumSetRef
Function overview	<p>This function is used to register or reference service setting number data. Up to 10 service setting number data items can be registered.</p> <p>In registration, specify the service setting number data and table number. In reference, specify the service setting number and reference one by one data.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeSrvInfoNumSetRef(mode, no, data);		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
no	unsigned char	I	Specification table number (0 to 9)
Data	_MSL_SRV_SET_INFO *	I/O	Pointer of the service setting number data storage area (*1)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (parameter error)
Remarks	<p>*1</p> <p>[Service setting number structure]</p> <pre>typedef struct tag_MSL_SRV_SET_INFO{ unsigned char Flg ; /* Identifier flag */ /* 0: No data registered */ /* 1: Special number */ /* 2: USSD code number */ char Title[MSL_SRVINFO_TITLE] ; /* Service setting name character */ }string</pre>		

```

    char    Send_No[MSL_SRVINFO_SENDNO] ;/* Dial data          <ASC> */
} _MSL_SRV_SET_INFO ;

#define MSL_SRVINFO_TITLE  20      /* Number of service setting name
character strings          */
#define MSL_SRVINFO_SENDNO 40      /* Number of dial data character
strings                    */

<Definition of Flg>
MSL_SRV_DATANON:  No data registered
MSL_SRV_TOKUBAN:  Special number
MSL_SRV_USSDCODE:  USSD code

```

11.54 Deleting Service Setting Number

Classification	General function		
Function name	Deleting service setting number	Symbol	Msl_OpeSrvInfoNumDelete
Function overview	<p>This function is used to delete one or all service setting number data items having the specified table numbers.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeSrvInfoNumDelete(no, kind);		
Argument name	Type	I/O	Explanation
no	unsigned char	I	Specification table number (0 to 9)
kind	unsigned char	I	Type: MSL_SRVDEL: Deletes one data item. MSL_SRVDELALL: Deletes all data items.
Return value	Type	I/O	Explanation
ret	Int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (parameter error)
Remarks	<p>Up to 10 data items can be registered.</p> <p>When specifying MSL_SRVDELALL, no is ignored..</p> <p>Use a null to initialize a character string and dial of the service setting name.</p> <p>Specify MSL_SRV_DATANON for the identifier.</p> <p>When the identifier is MSL_SRV_DATANON, the system assumes that there is no data, but returns MSL_OK for the processing result.</p> <p>After deletion, table reconstruction is not performed.</p>		

11.55 Setting or Referencing IMEI

Classification	General function		
Function name	Setting or referencing IMEI	Symbol	Msl_OpeIMEISetRef
Function overview	<p>This function is used to set or reference IMEI.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeIMEISetRef (mode, imei)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
imei	unsigned char *	I/O	Stores the data value to be set or referenced. IMEI value
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (parameter error)
Remarks			

11.56 Setting or Referencing Accumulated Reset Date and Time

Classification	Service function		
Function name	Setting or referencing accumulated reset date and time	Symbol	Msl_OpeTalkRstDateSetRef
Function overview			
This function is used to set or reference the accumulated reset data and time.			
Include file	res_ope.h		
Calling sequence	int Msl_OpeTalkRstDateSetRef (mode, data);		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
data	_MSL_OPE_RSTDATE *	I/O	Pointer of the area for storing accumulated reset date setting data (*1)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (parameter error)
Remarks			
<div>*1</div> <div>[Accumulated reset date setting structure]</div> <div>typedef struct tagMSL_OPE_RSTDATE</div> <div>{</div> <div> unsigned char Month; /* Month<BIN> */</div> <div> unsigned char Day ; /* Day<BIN> */</div> <div> unsigned char Hour ; /* Hour<BIN> */</div> <div> unsigned char Min ; /* Minute<BIN> */</div> <div>} _MSL_OPE_RSTDATE ;</div>			

11.57 Setting or Referencing Response Message

Classification	General function		
Function name	Setting or referencing response message	Symbol	Msl_OpeSrvRcvMsgInfoSetRef
Function overview			
This function is used to set or reference response message data.			
Include file	res_ope.h		
Calling sequence	int Msl_OpeSrvRcvMsgInfoSetRef(mode , num, data)		
Argument name	Type	I/O	Explanation
mode	Int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
num	unsigned char	I	Specification table number (0 to 9)
data	_MSL_SRV_RCVMSG_SET_INFO *	I/O	Pointer of the response message data storage area (*1)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (parameter error)
Remarks			
<div>*1</div> <div>[Response message data]</div> <div>typedef struct tagMSL_SRV_RCVMSG_SET_INFO</div> <div>{</div> <div> unsigned char Rcv_Msg[MSL_SRVRCVMSG_DATA] ; /* Response message reception data */</div> <div> unsigned char Title[MSL_SRVRCVMSG_TITLE] ; /* Response message name */</div> <div>} _MSL_SRV_RCVMSG_SET_INFO ;</div> <div>#define MSL_SRVRCVMSG_DATA 20 /* Maximum size of response message reception data */</div> <div>#define MSL_SRVRCVMSG_TITLE 20 /* Maximum size of the response message name */</div> <div>#define MSL_SRVRCVMSGDATA_MAX 10 /* Maximum number of response messages</div>			

registered	*/
------------	----

11.58 Deleting Response Message

Classification	General function		
Function name	Deleting response message	Symbol	Msl_OpeSrvRcvMsgInfoDelete
Function overview			
This function is used to delete response message data.			
Include file	res_ope.h		
Calling sequence	int Msl_OpeSrvRcvMsgInfoDelete(num, kind)		
Argument name	Type	I/O	Explanation
num	unsigned char	I	Specification table number (0 to 9)
kind	unsigned char	I	Deletion type MSL_SRVDEL: Deletes on data item. MSL_SRVDELALL: Deletes all items.
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (parameter error)
Remarks			
#define MSL_SRVRVMSGDATA_MAX 10 /* Maximum number of response messages registered */			

11.59 Setting or Referencing Camera Shooting Image Size

Classification	General function		
Function name	Setting or referencing camera shooting image size	Symbol	Msl_OpeCam_SizeSetRef
Function overview			
This function is used to set or reference the information about the size of the image shot with the camera.			
Include file	res_ope.h		
Calling sequence	int Msl_OpeCam_SizeSetRef (mode, style, rec_mode, size)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies mode. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
style	int	I	Specifies a camera. MSL_FRONT: Front camera MSL_SIDE: Side camera
rec_mode	int	I	Recording mode MSL_CAM_RECSIZE_SNAP: Snap MSL_CAM_RECSIZE_MOVE: Recording mode MSL_CAM_RECSIZE_RAP: Continuous shooting mode
size	int *	I/O	Image size data (*1) MSL_CAM_SIZE_KABE: Wallpaper MSL_CAM_SIZE_VGA: VGA MSL_CAM_SIZE_QVGA: QVGA MSL_CAM_SIZE_CIF: CIF MSL_CAM_SIZE_QCIF: QCIF MSL_CAM_SIZE_SQCIF: sub-QCIF MSL_CAM_SIZE_UXGA: UXGA MSL_CAM_SIZE_SXGA: SXGA
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (parameter error)
Remarks			
*1 Set camera image size data.			



11.60 Setting or Referencing White Balance

Classification	General function		
Function name	Setting or referencing white balance	Symbol	Msl_OpeCam_WhiteBalanceSetRef
Function overview			
This function is used to set or reference the camera white balance.			
Include file	res_ope.h		
Calling sequence	int Msl_OpeCam_WhiteBalanceSetRef (mode, value)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies mode. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
value	int *	I/O	Sets the white balance (*1). MSL_CAM_WB_FINE: Fine MSL_CAM_WB_CLOUDY: Cloudy MSL_CAM_WB_AUTO: Auto MSL_CAM_WB_ELECTRIC: Electric light
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (parameter error)
Remarks			
*1 Set the camera white balance.			

11.61 Setting or Referencing Various Setting Information Items about Camera Function 2

Classification	General function		
Function name	Setting or referencing various setting information items about camera function 2	Symbol	Msl_OpeCameraInfSetRef2
Function overview			
This function is used to set or reference the information about the size of the image shot with the camera.			
Include file	res_ope.h		
Calling sequence	int Msl_OpeCameraInfSetRef2 (mode, data)		
Argument name	Type	I/O	Explanation
mode	unsigned char	I	Specifies mode. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
data	_MSL_OPE_GENERALCONF2 *	I/O	Data to be set or referenced (*1)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (parameter error)
Remarks			
<div>*1 Various camera information structure 2</div> <div>typedef struct tagMSL_OPE_GENERALCONF2</div> <div>{</div> <div> unsigned char SelfTimer ; /* Number of seconds for the selftimer */</div> <div> unsigned char ShutSound_ST; /* Shutter sound for still image or continuous shooting */</div> <div> unsigned char ShutSound_MV; /* Shutter sound for animation */</div> <div> unsigned char BurstMode; /* Continuous shooting type auto or manual */</div> <div> unsigned char AutoKeep; /* Automatic save */</div> <div> unsigned char SheetCount; /* Number of continuous shooting pictures */</div> <div> unsigned char DispSize; /* Display size */</div> <div> unsigned char Reserve; /* Reserved */</div> <div> } _MSL_OPE_GENERALCONF2 ;</div>			

11.62 Setting or Referencing MPEG-4 Encoder Parameters 2

Classification	General function		
Function name	Setting or referencing MPEG-4 encoder parameters 2	Symbol	Msl_OpeMP4EncParamSetRef2
Function overview			
This function is used to set or reference the MPEG-4 encoder parameters.			
Include file	res_ope.h		
Calling sequence	int Msl_OpeMP4EncParamSetRef2 (mode, sizemode, data)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies mode. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
sizemode	unsigned char	I/O	Image size mode MSL_MOVSQCIFTIME: Animation recording, SubQCIF, time priority MSL_MOVSQCIFNORM: Animation recording, SubQCIF, standard MSL_MOVSQCIFIMG: Animation recording, SubQCIF, image quality priority MSL_MOVQCIFTIME: Animation recording, QCIF, time priority MSL_MOVQCIFNORM: Animation recording, QCIF, standard MSL_MOVQCIFIMG: Animation recording, QCIF, image quality priority MSL_TV64CAMQCIFMOV: TV phone 64k, camera, QCIF, motion priority MSL_TV64CAMQCIFNORM: TV phone 64k, camera, QCIF, standard MSL_TV64CAMQCIFIMG: TV phone 64k, camera, QCIF, image quality priority

		<p>MSL_TV64PICQCIFMOV: TV phone 64k, still image, QCIF, motion priority</p> <p>MSL_TV64PICQCIFNORM: TV phone 64k, still image, QCIF, standard</p> <p>MSL_TV64PICQCIFIMG: TV phone 64k, still image, QCIF, image quality priority</p> <p>MSL_TV32CAMQCIFMOV: TV phone 32k, QCIF, camera, motion priority</p> <p>MSL_TV32CAMQCIFNORM: TV phone 32k, QCIF, camera, standard</p> <p>MSL_TV32CAMQCIFIMG: TV phone 32k, QCIF, camera, image quality priority</p> <p>MSL_TV32PICQCIFMOV: TV phone 32k, QCIF, still image, motion priority</p> <p>MSL_TV32PICQCIFNORM: TV phone 32k, QCIF, still image, standard</p> <p>MSL_TV32PICQCIFIMG: TV phone 32k, QCIF, still image, image quality priority</p> <p>MSL_PICTUREVOICE_MP4ENC: Picture voice MP4 encode</p> <p>MSL_PICTUREVOICE_AMRENC: Picture voice AMR encode</p> <p><VDCI-VOP setting information></p> <p>MSL_VDCSQCIFTIME: Animation recording (SubQCIF, time priority)</p> <p>MSL_VDCSQCIFNORM: Animation recording (SubQCIF, standard)</p> <p>MSL_VDCSQCIFIMG: Animation recording (SubQCIF, image quality priority)</p> <p>MSL_VDCQCIFTIME: Animation recording (QCIF, time priority)</p> <p>MSL_VDCQCIFNORM: Animation recording (QCIF, standard)</p> <p>MSL_VDCQCIFIMG: Animation recording (QCIF, image quality)</p>
--	--	--

			<p>priority)</p> <p>MSL_TV64VDC: TV64K</p> <p>MSL_TV32VDC: TV32K</p> <p><DSP Enc encoding rate></p> <p>MSL_DSPENCSRATE_VCSI: VCSI</p> <p>MSL_DSPENCSRATE_IF2M32K: For TV phone 32K</p> <p>MSL_DSPENCSRATE_IF2M64K: For TV phone 64K</p> <p>MSL_DSPENCSRATE_STSTIME: Animation recording (SubQCIF, time priority)</p> <p>MSL_DSPENCSRATE_STSNORM: Animation recording (SubQCIF, standard)</p> <p>MSL_DSPENCSRATE_STSIMG: Animation recording (SubQCIF, image quality priority)</p> <p>MSL_DSPENCSRATE_STQTIME: Animation recording (QCIF, time priority)</p> <p>MSL_DSPENCSRATE_STQNORM: Animation recording (QCIF, standard)</p> <p>MSL_DSPENCSRATE_STQIMG: Animation recording (QCIF, image quality priority)</p> <p><H.263 Annex mode></p> <p>MSL_VDSPH263ANNEX_MOV: VDSP encoder H.263 Annex mode (animation recording)</p> <p>MSL_VDSPH263ANNEX_TV: VDSP encoder H.263 Annex mode (TV phone)</p> <p><I-VOP quantization step size></p> <p>MSL_VDSPSTEPSIZE_PICV: VDSP encoder I-VOP quantization step size (Picture Voice)</p> <p>MSL_VDSPSTEPSIZE_TV: VDSP encoder I-VOP quantization step size (TV phone)</p>
--	--	--	---

		<p><Encode mode></p> <p>MSL_VDSPENC_MOV: Encode mode (animation recording)</p> <p>MSL_VDSPENC_TV: Encode mode (TV phone)</p> <p><Strike shot></p> <p>MSL_STRKSHOT_SPACE: I-Frame insertion time interval</p> <p>MSL_STRKSHOT_STEPSIZE_SQCIFTIME: VOP quantization step size (animation recording, SubQCIF, time priority)</p> <p>MSL_STRKSHOT_STEPSIZE_SQCIFNORM: VOP quantization step size (animation recording, SubQCIF, standard)</p> <p>MSL_STRKSHOT_STEPSIZE_SQCIFIMG: VOP quantization step size (animation recording, SubQCIF, image quality priority)</p> <p>MSL_STRKSHOT_STEPSIZE_QCIFTIME: VOP quantization step size (animation recording, QCIF, time priority)</p> <p>MSL_STRKSHOT_STEPSIZE_QCIFNORM: VOP quantization step size (animation recording, QCIF, standard)</p> <p>MSL_STRKSHOT_STEPSIZE_QCIFIMG: VOP quantization step size (animation recording, QCIF, image quality priority)</p> <p>MSL_VDCSLICE_LEN_TV64K: Slice length (TV phone 64k)</p> <p>MSL_VDCSLICE_LEN_TV32K: Slice length (TV phone 32k)</p> <p>MSL_VDCCYCLE_REFRESH_TV64K: Cycle refresh (TV phone 64k)</p> <p>MSL_VDCCYCLE_REFRESH_TV32K: Cycle</p>
--	--	---

			refresh (TV phone 32k) MSL_VDCADJUST_INTRAREFRESH_TV64K Adaptive intra refresh (TV phone 64k) MSL_VDCADJUST_INTRAREFRESH_TV32K Adaptive intra refresh (TV phone 32k)
data	unsigned char *	I/O	MPEG-4 encoder parameter storage area
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK : Normal end MSL_NG : Abnormal end (parameter error)
Remarks	<p><Image size mode> <EEPROM (ORG) area></p> <ul style="list-style-type: none"> - MSL_MOVSQCIFTIME: Animation recording, SubQCIF, time priority => Page31C address 0 to 4 - MSL_MOVSQCIFNORM: Animation recording, SubQCIF, standard => Page31C address 5 to 9 - MSL_MOVSQCIFIMG: Animation recording, SubQCIF, image quality priority => Page31C address A to E - MSL_MOVQCIFTIME: Animation recording, QCIF, Time priority => Page31D address 0 to 4 - MSL_MOVQCIFNORM: Animation recording, QCIF, standard => Page31D address 5 to 9 - MSL_MOVQCIFIMG: Animation recording, QCIF, image quality priority => Page31D address A to E - MSL_TV64CAMQCIFMOV: TV phone 64k, camera, QCIF, motion priority => Page31E address 0 to 4 - MSL_TV64CAMQCIFNORM: TV phone 64k, camera, QCIF, standard => Page31E address 5 to 9 - MSL_TV64CAMQCIFIMG: TV phone 64k, camera, QCIF, image quality priority => Page31E address A to E - MSL_TV64PICQCIFMOV: TV phone 64k, still image, QCIF, motion priority => Page31F address 0 to 4 - MSL_TV64PICQCIFNORM: TV phone 64k, still image, QCIF, standard => Page31F address 5 to 9 - MSL_TV64PICQCIFIMG: TV phone 64k, still image, QCIF, image quality priority => Page31F address A to E - MSL_TV32CAMQCIFMOV: TV phone 32k, QCIF, camera, motion priority => Page320 address 0 to 4 - MSL_TV32CAMQCIFNORM: TV phone 32k, QCIF, camera, standard => Page320 address 5 to 9 - MSL_TV32CAMQCIFIMG: TV phone 32k, QCIF, camera, image quality priority => Page320 address A to E - MSL_TV32PICQCIFMOV: TV phone 32k, QCIF, still image, motion priority => Page321 address 0 to 4 		

- MSL_TV32PICQCIFNORM: TV phone 32k, QCIF, still image, standard	=>	
Page321 address 5 to 9		
- MSL_TV32PICQCIFIMG: TV phone 32k, QCIF, still image, image quality priority	=>	
Page321 address A to E		
- MSL_PICTUREVOICE_MP4ENC: Picture voice MP4 encode	=>	
Page322 address 0 to 4		
- MSL_PICTUREVOICE_AMRENC: Picture voice AMR encode	=>	
Page307 address 9		
<VDCI-VOP setting information>		
- MSL_VDCSQCIFTIME: Animation recording (SubQCIF, time priority)	=>	Page304
address 0 to 1		
- MSL_VDCSQCIFNORM: Animation recording (SubQCIF, standard)	=>	Page304
address 2 to 3		
- MSL_VDCSQCIFIMG: Animation recording (SubQCIF, image quality priority)	=>	
Page304 address 4 to 5		
- MSL_VDCQCIFTIME: Animation recording (QCIF, time priority)	=>	Page304
address 6 to 7		
- MSL_VDCQCIFNORM: Animation recording (QCI,F, standard)	=>	Page304
address 8 to 9		
- MSL_VDCQCIFIMG: Animation recording (QCIF, image quality priority)	=>	
Page304 address A to B		
- MSL_TV64VDC: TV64K	=>	Page304
address C to D		
- MSL_TV32VDC: TV32K	=>	Page304
address E to F		
<DSP Enc encoding rate>		
- MSL_DSPENCSRATE_VCSI: Encoding rate (VCSI)	=>	Page307 address 0
- MSL_DSPENCSRATE_IF2M32K: Encoding rate (TV phone 32K)	=>	Page307
address 1		
- MSL_DSPENCSRATE_IF2M64K: Encoding rate (TV phone 64K)	=>	Page307
address 2		
- MSL_DSPENCSRATE_STSTIME: Encoding rate (animation recording (SubQCIF, time priority))		
	=>	Page307 address 3
- MSL_DSPENCSRATE_STSNORM: Encoding rate (animation recording (SubQCIF, standard))		
	=>	Page307 address 4
- MSL_DSPENCSRATE_STSIMG: Encoding rate (animation recording (SubQCIF, image quality priority))		
	=>	Page307 address 5
- MSL_DSPENCSRATE_STQTIME: Encoding rate (animation recording (QCIF, time priority))		
	=>	Page307 address 6

- MSL_DSPENCSTRATE_STQNORM: Encoding rate (animation recording (QCIF, standard))
=> Page307 address 7
- MSL_DSPENCSTRATE_STQIMG: Encoding rate (animation recording (QCIF, image quality priority))
=> Page307 address 8
- <VDSP encoder H.263 Annex mode>
- MSL_VDSPH263ANNEX_MOV: VDSP encoder H.263 Annex mode (animation recording)
=> Page305 address 0
- MSL_VDSPH263ANNEX_TV: VDSP encoder H.263 Annex mode (TV phone) => Page305 address 1
- <VDSP encoder I-VOP quantization step size>
- MSL_VDSPSTEPSIZE_PICV: VDSP encoder I-VOP quantization step size (Picture Voice)=> Page305 address 9
- MSL_VDSPSTEPSIZE_TV: VDSP encoder I-VOP quantization step size (TV phone) => Page305 address 8
- <Encode mode>
- MSL_VDSPENC_MOV: Encode mode (animation recording) => Page306 address 0
- MSL_VDSPENC_TV: Encode mode (TV phone) => Page306 address 1
- <Strike shot>
- MSL_STRKSHOT_SPACE: At strike shot, I-Frame insertion time interval => Page36A address 0
- MSL_STRKSHOT_STEPSIZE_SQCIFTIME: I-VOP quantization step size (animation recording, SubQCIF, time priority)
=> Page305 address 2
- MSL_STRKSHOT_STEPSIZE_SQCIFNORM: I-VOP quantization step size (animation recording, SubQCIF, standard)
=> Page305 address 3
- MSL_STRKSHOT_STEPSIZE_SQCIFIMG: I-VOP quantization step size (animation recording, SubQCIF, image quality priority)
=> Page305 address 4
- MSL_STRKSHOT_STEPSIZE_QCIFTIME: I-VOP quantization step size (animation recording, QCIF, time priority)
=> Page305 address 5
- MSL_STRKSHOT_STEPSIZE_QCIFNORM: I-VOP quantization step size (animation recording, QCIF, standard)
=> Page305 address 6
- MSL_STRKSHOT_STEPSIZE_QCIFIMG: I-VOP quantization step size (animation recording, QCIF, image quality priority)
=> Page305 address 7

- MSL_VDCSLICE_LEN_TV64K: Slice length (TV phone 64k)
=> Page306 address 2 to 3
- MSL_VDCSLICE_LEN_TV32K: Slice length (TV phone 32k)
=> Page306 address 4 to 5
- MSL_VDCCYCLE_REFRESH_TV64K: Cycle refresh (TV phone 64k)
=> Page306 address 6
- MSL_VDCCYCLE_REFRESH_TV32K: Cycle refresh (TV phone 32k)
=> Page306 address 7
- MSL_VDCADJUST_INTRAREFRESH_TV64K: Adaptive intra refresh (TV phone 64k)
=> Page306 address 8
- MSL_VDCADJUST_INTRAREFRESH_TV32K: Adaptive intra fresh (TV phone 32k)
=> Page306 address 9

11.63 Setting or Referencing Encoder parameters for Recording Animations

Classification	General function		
Function name	Setting or referencing encoder parameters for recording animations	Symbol	Msl_OpeMovieEncParamSetRef
Function overview	This function is used to set or reference the encoder parameters at recording animations.		
Include file	res_ope.h		
Calling sequence	int Msl_OpeMovieEncParamSetRef (mode, quality, resolution, staticparam, dynamicparam);		
Argument name	Type	I/O	Explanation
Mode	int	I	Specifies mode. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
Quality	int	I	Sets recording image quality. MSL_VREC_ASF_SFINE: Superfine MSL_VREC_ASF_FINE: Fine MSL_VREC_ASF_NOMAL: Normal MSL_VREC_MP4_FINE: High quality MSL_VREC_MP4_NOMAL: Standard MSL_VREC_MP4_ECN: Long-time
Resolution	int	I	Specifies a size. MSL_SUBSQCIF : SubQCIF MSL_QCIF : QCIF MSL_QVGA : QVGA
Staticparam	MslVdspEncStaticParam_p *	I/O	Encode parameter (static) (*1)
Dynamicparam	MslVdspEncDynamicParam_p *	I/O	Encode parameter (dynamic) (*2)
Return value	Type	I/O	Explanation
Ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (parameter error)
Remarks	<p>*1 Structure that stores the encoder parameters (that can be specified only at start)</p> <pre>typedef struct tagMslVdspEncStaticParam_p { MslVdspEncStaticParamFlags flags; /* Effective member */ }</pre>		

```

    MslVdspEncRateMode    ratemode;          /* Rate control method          */
    MslVdspEncPath        path;              /* Data path                    */
    MslVdspEncReset       reset;             /* Whether to reset            */
    MslVdspEncStartLayer  esl;               /* Encoder start layer         */
    MslVdspCodec          en;                /* Encode method               */
    MslVdspEncFormat_p    format;            /* Format                       */
    MslVdspEncDu          du;                /* AL-SDU align mode           */
    MslVdspEncPictureType pt;                /* Encode start picture        */
    MslVdspEncAnnexMode    annexmode;        /* H.263 Annex Mode           */
} MslVdspEncStaticParam_p;

*2 Structure that stores the encoder parameters (that can always be specified)
typedef struct tagMslVdspEncDynamicParam_p
{
    MslVdspEncDynamicParamFlags flags;        /* Flag indicating valid or invalid */
    MslVdspEncFrameRate    framerate;        /* Frame rate specification          */
    MslVdspEncBitRate      bitrate;          /* Bit rate specification            */
    unsigned int           sl;               /* Slice length                      */
    unsigned int           cir;              /* Cyclic Refresh                    */
    unsigned int           air;              /* Adaptive Refresh                  */
    unsigned int           rt;               /* Refresh Time                      */
    unsigned int           qt_max;           /* Setting upper limit of quantization */
    unsigned int           qt_min;           /* Setting lower limit of quantization */
    MslVdspEncMs           ms;               /* Moving vector search range        */
    unsigned int           ivop_qt;          /* I-VOP quantization value          */
    unsigned int           ivop;             /* I-VOP interval                    */
    unsigned int           storage_ivop_qt;  /* Storage-CBR default quantization value */
    unsigned int           storage_ivop;     /* Storage-CBR I-VOP interval        */
    MslBoolean            storage_sv;        /* Storage-CBR still VOP encoding function */
    MslBoolean            storage_vfr;       /* Storage-CBR VFR function          */
} MslVdspEncDynamicParam_p;

typedef unsigned int MslBoolean;          /* True/False */
typedef unsigned int MslVdspEncDynamicParamFlags; /* Specifiable AL-SDU */
typedef unsigned int MslVdspEncFrameRate; /* Frame rate */
typedef unsigned int MslVdspEncBitRate;    /* Bit rate */
typedef unsigned int MslVdspEncMs;         /* Moving vector search range */
typedef unsigned int MslVdspEncStaticParamFlags; /* Parameter flag */
typedef unsigned int MslVdspEncRateMode;   /* Rate control method */
typedef unsigned int MslVdspEncPath;       /* Data utilization method */
typedef unsigned int MslVdspEncReset;      /* Whether to reset */
typedef unsigned int MslVdspEncStartLayer; /* Encode start layer */
typedef unsigned int MslVdspCodec;         /* Specifiable codec */
typedef unsigned int MslVdspEncDu;         /* Specifiable AL-SDU */
typedef unsigned int MslVdspEncPictureType; /* PictureType */

```

```
typedef      unsigned      int      MslVdspEncAnnexMode;          /*      H.263
AnnexMode          */

/***** Encoder parameter format structure *****/
typedef struct tagMslVdspEncFormat_p
{
    unsigned int  left ;
    unsigned int  top  ;
    unsigned int  right ;
    unsigned int  bottom ;
} MslVdspEncFormat_p;
```


11.64 Setting or Recording Replay Animation Display Method

Classification	General function		
Function name	Setting or referencing replay animation display method	Symbol	Msl_OpeMovieDispSizeSetRef
Function overview	<p>This function is used to set or reference the state of displaying the replay animation at the same magnification or zoom.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeMovieDispSizeSetRef (mode, value)		
Argument name	Type	I/O	Explanation
Mode	Int	I	Specifies mode. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
Value	int *	I/O	Replay animation display method MSL_MOVIE_DISP_NORMAL: Display at the same magnification MSL_MOVIE_DISP_LARGE: Zoom
Return value	Type	I/O	Explanation
Ret	Int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (parameter error)
Remarks			

11.65 Setting or Referencing Prefix

Classification	General function		
Function name	Setting or referencing prefix	Symbol	Msl_OpePrefixSetRef
Function overview	This function is used to set or reference a prefix.		
Include file	res_ope.h		
Calling sequence	int Msl_OpePrefixSetRef (mode, no, predata)		
Argument name	Type	I/O	Explanation
Mode	int	I	Specifies mode. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
No	unsigned char	I/O	Specifies a prefix. MSL_PREFIX1: Prefix 1 MSL_PREFIX2: Prefix 2 MSL_PREFIX3: Prefix 3 MSL_PREFIX4: Prefix 4 MSL_PREFIX5: Prefix 5 MSL_PREFIX6: Prefix 6 MSL_PREFIX7: Prefix 7
Predate	_MSL_OPE_PREFIX *	I/O	Prefix data storage area (*1)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (parameter error)
Remarks	<p>*1 Prefix information structure</p> <pre>typedef struct tagMSL_OPE_PREFIX { unsigned char name[MSL_PREFIXNAMEMAX+1]; /* Prefix name */ unsigned char no[MSL_PREFIXNOMAX+1]; /* Prefix number */ } _MSL_OPE_PREFIX;</pre> <pre>#define MSL_PREFIXNAMEMAX 16 /* Maximum prefix name size */</pre>		

```
#define MSL_PREFIXNOMAX    10                /* Maximum prefix number size */
```

11.66 Setting or Referencing Camera or Download Folder Name

Classification	General function		
Function name	Setting or referencing camera or download folder name	Symbol	Msl_OpeFolderMakeSetRef
Function overview	<p>This function is used to set or reference a name of the camera or download folder that the user can be specified.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeFolderMakeSetRef (mode, folkind, no, fldname)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies mode. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
folkind	unsigned char	I	Folder type (camera or download) MSL_OPE_FOL01: Camera MSL_OPE_FOL02: Download
no	unsigned char	I	Folder number (1 to 9)
fldname	unsigned char *	I/O	Folder name storage area pointer (*1)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (out of range of folder numbers)
Remarks	<p>*1 Up to 18 bytes can be used for the folder name.</p>		

11.67 Setting or Referencing MPEG4 Capability

Classification	General function		
Function name	Setting or referencing MPEG4 capability	Symbol	Msl_OpeTVTeI_Mpeg4CodecSetRef
Function overview			
This function is used to set or reference whether the video encoding system has the MPEG4 capability.			
Include file	res_ope.h		
Calling sequence	int Msl_OpeTVTeI_Mpeg4CodecSetRef(mode, status)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies mode. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
status	int *	I/O	Video encoding system MSL_TVTEL_MPEG4CODEC_ON: With MPEG4 MSL_TVTEL_MPEG4CODEC_OFF: Without MPEG4
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			
The default value is “with MPEG4.”			

11.68 Setting or Referencing Terminal Manufacturer

Classification	General function		
Function name	Setting or referencing terminal manufacturer	Symbol	Msl_OpeTVTel_TermMakerSetRef
Function overview			
This function is used to set or reference terminal manufacturer information.			
Include file	res_ope.h		
Calling sequence	int Msl_OpeTVTel_TermMakerSetRef(mode, status)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies mode. MSL_FUNCREF: Reference processing
status	int *	I/O	Terminal manufacturer information MSL_TVTEL_MAKER_PANA: Panasonic MSL_TVTEL_MAKER_NEC: NEC
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			
The default value is “NEC.”			

11.69 Setting or Referencing H.223 Level

Classification	General function		
Function name	Setting or referencing H.223 level	Symbol	Msl_OpeTVTel_H223LevelSetRef
Function overview			
This function is used to set or reference the H.223 level.			
Include file	res_ope.h		
Calling sequence	int Msl_OpeTVTel_H223LevelSetRef(mode, status)		
Argument name	Type	I/O	Explanation
Mode	Int	I	Specifies mode. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
Status	int *	I/O	H.223 level MSL_TVTEL_H223LV_0 : 0 MSL_TVTEL_H223LV_1 : 1 MSL_TVTEL_H223LV_2 : 2
Return value	Type	I/O	Explanation
Ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			
- The default value of the H.223 level is “MSL_TVTEL_H223LV_2.”			

11.70 Setting or Referencing Reciprocating Delay Time

Classification	General function		
Function name	Setting or referencing reciprocating delay time	Symbol	Msl_OpeTVTel_RoundTripDelay
Function overview			
This function is used to set or reference the reciprocating delay time.			
Include file	res_ope.h		
Calling sequence	int Msl_OpeTVTel_RoundTripDelay(mode, time)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies mode. MSL_FUNCREF: Reference processing
time	int *	I/O	Reciprocating delay time (msec)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			
- The default value of the reciprocating delay time is “1140.”			

11.71 Setting or Referencing Video Encoding Parameters

Classification	General function		
Function name	Setting or referencing video encoding parameters	Symbol	Msl_OpeTVTel_VideoEncParamSetRef
Function overview			
This function is used to set or reference the video encoding control parameters.			
Include file	res_ope.h		
Calling sequence	int Msl_OpeTVTel_VideoEncParamSetRef(mode, encparam)		
Argument name	Type	I/O	Explanation
mode	Int	I	Specifies mode. MSL_FUNCREF: Reference processing
encparam	_MSL_OPEVENCPARAM *	I/O	Data to be set or referenced Pointer of the video encoding parameter array table. (See Remarks for details.)
Return value	Type	I/O	Explanation
ret	Int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			
Details of the video encoding parameter array table MSL_OPEVENDPARAM encparam [4][3]; typedef struct tagMSL_OPEVENCPARAM { unsigned int framerate ; /* Frame rate */ unsigned int bitrate ; /* Bit rate */ unsigned int sl ; /* Slice length */ unsigned int qt_max ; /* Upper limit of quantization */ unsigned int qt_min ; /* Lower limit of quantization */ unsigned int air ; /* AdaptiveIntraRefresh */ unsigned int rt ; /* RefreshTime */ unsigned int cir; /* CyclicIntraRefresh */ }_MSL_OPEVENCPARAM ;			

Explanation of array numbers

- Primary array[3]: Image quality setting

- 0 /* Motion priority */
- 1 /* Standard */
- 2 /* Image quality priority */

- Secondary array[4]: VIDEO bit rate (bps)

- 0 /* 16K */
- 1 /* 24K */
- 2 /* 48K */
- 3 /* 56K */

Define values (Use the following define values for the frame rate and bit rate.)

- Frame rate Define value

MSL_TVTEL_FR_1HZ :1Hz	0x00000000
MSL_TVTEL_FR_2HZ :2Hz	0x00000001
MSL_TVTEL_FR_3HZ :3Hz	0x00000002
MSL_TVTEL_FR_5HZ :5Hz	0x00000003
MSL_TVTEL_FR_6HZ :6Hz	0x00000004
MSL_TVTEL_FR_7POINT5HZ :7.5Hz	0x00000005
MSL_TVTEL_FR_10HZ :10Hz	0x00000006
MSL_TVTEL_FR_15HZ :15Hz	0x00000007
MSL_TVTEL_FR_30HZ :30Hz	0x00000008

- Bit rate Define value

MSL_TVTEL_BR_16KBPS :16Kbps	0x00000000
MSL_TVTEL_BR_24KBPS :24Kbps	0x00000001
MSL_TVTEL_BR_32KBPS :32Kbps	0x00000002
MSL_TVTEL_BR_40KBPS :40Kbps	0x00000003
MSL_TVTEL_BR_48KBPS :48Kbps	0x00000004
MSL_TVTEL_BR_56KBPS :56Kbps	0x00000005
MSL_TVTEL_BR_64KBPS :64Kbps	0x00000006
MSL_TVTEL_BR_128KBPS :128Kbps	0x00000007
MSL_TVTEL_BR_192KBPS :192Kbps	0x00000008
MSL_TVTEL_BR_256KBPS :256Kbps	0x00000009
MSL_TVTEL_BR_320KBPS :320Kbps	0x0000000A
MSL_TVTEL_BR_384KBPS :384Kbps	0x0000000B

11.72 Setting or Referencing QOS Parameters

Classification	General function		
Function name	Setting or referencing QOS parameters	Symbol	Msl_OpeTVTeI_QosParamSetRef
Function overview			
This function is used to set or reference the parameters at QOS reception.			
Include file	res_ope.h		
Calling sequence	int Msl_OpeTVTeI_QosParamSetRef(mode, qosparam)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies mode. MSL_FUNCREF: Reference processing
qosparam	_MSL_OPEQOSPARAM *	I/O	Data to be set or referenced Pointer of the QOS reception parameter array table (See Remarks for details.)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			
Details of the video encoding parameter array table MSL_OPEQOSPARAM qosparam [3][15]; typedef struct tagMSL_OPEQOSPARAM { unsigned int air ; /* Adaptive Intra Refresh */ unsigned int cir ; /* Cyclic Intra Refresh */ unsigned int rt ; /* Refresh Time */ }_MSL_OPEQOSPARAM ; Explanation of array numbers - Primary array[15]: Packet loss rate range (in units of 0.1%) 0 /* 0 */ 1 /* 1 */ 2 /* 2 */ 3 /* 3 */			

```
4      /* 4 */
5      /* 5 */
6      /* 6 */
7      /* 7 */
8      /* 8 */
9      /* 9 */
10     /* 10~19 */
11     /* 20~49 */
12     /* 50~99 */
13     /* 100~199 */
14     /* 200~1000 */
```

- Secondary array[3]: Image quality setting

```
0      /* Motion priority */
1      /* Standard */
2      /* Image quality priority */
```

11.73 Setting or Referencing Image Display Method

Classification	General function		
Function name	Setting or referencing image display method	Symbol	Msl_Opelmg_DispModeSetRef
Function overview			
This function is used to set or reference the image viewer image display method.			
Include file	res_ope.h		
Calling sequence	int Msl_Opelmg_DispModeSetRef (mode, dispmode)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
dispmode	int *	I/O	Image display method MSL_IMG_DISP_NORMAL: Standard MSL_IMG_DISP_SIZECHANGE: Display in screen size
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			

11.74 Setting or Referencing Individual Information about Communication Data

Classification	General function		
Function name	Setting or referencing individual information about communication data	Symbol	Msl_OpeComDataSetRef
Function overview			
<p>This function is used to set or reference individual information about communication data.</p> <p>* The data storage areas are as follows:</p> <p>File system: Site names 1 to 20, ESSID setting 1 to 20</p> <p>Nonvolatile data: Data that is not stored in the file system</p>			
Include file	res_ope.h		
Calling sequence	int Msl_OpeComDataSetRef(mode, func_id, data)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
func_id	int	I	Specified function number MSL_SITENAME1: Site name 1 (profile name 1) ~ ~ MSL_SITENAME20: Site name 20 (profile name 20) MSL_ESSIDSET1: ESSID setting 1 ~ ~ MSL_ESSIDSET20: ESSID setting 20 MSL_WAIT_POWERSAVE: At standby (power save setting) MSL_COM_POWERSAVE: At communication (power save setting) MSL_184DISPLAYNAME: 184 terminal display names MSL_REGISTEREXPIRE: Register expire value MSL_SUBSCRIBEEXP: Subscribe expire value MSL_RTCPPACKET: RTCP packet transmission interval MSL_CONNECTWTIME: Connection wait time

data	void *	I/O	Pointer of storing each function data (*1)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			
*1			
The calling source must allocate an area having the size that can store the reference result for the pointer of storing each function data..			
<Conditions for setting or referencing each specified function number>			
- Function specification number - - Data name -			
MSL_SITENAME1 Site name 1 (profile name 1)			
~ ~			
MSL_SITENAME20 Site name 20 (profile name 20)			
MSL_ESSIDSET1 ESSID setting 1			
~ ~			
MSL_ESSIDSET20 ESSID setting 1			
MSL_WAIT_POWERSAVE At 184 standby (power save setting)			
MSL_COM_POWERSAVE At 184 communication (power save setting)			
MSL_184DISPLAYNAME 184 terminal display names			
MSL_REGISTEREXPIRE Register expire value			
MSL_SUBSCRIBEEXP Subscribe expire value			
MSL_RTCPPACKET RTCP packet transmission interval			
MSL_CONNECTWTIME Connection wait time			

11.75 Registering SD Power-off (SDM)

Classification	General function		
Function name	Registering SD power-off (SDM)	Symbol	Msl_OpeSdm_PowerOffSetRef
Function overview			
This function is used to set or reference information about the SD standard file API power-off.			
Include file	res_ope.h		
Calling sequence	int Msl_OpeSdm_PowerOffSetRef(mode, status)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
status	int *	I/O	STATUS decision
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			

11.76 Setting or Referencing SD-PIM Recovery Information

Classification	General function		
Function name	Setting or referencing SD-PIM recovery information	Symbol	Msl_OpeSDPim_RestoreSetRef
Function overview			
This function is used to set or reference SD-PIM file recovery data.			
Include file	res_ope.h		
Calling sequence	int Msl_OpeSDPim_RestoreSetRef(mode, encparam)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
encparam	_MSL_OPESDPPARAM *	I/O	Data to be set or referenced SD-PIM recovery information pointer (See Remarks for details.)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			
Details of SD-PIM recovery information _MSL_OPESDPPARAM encparam; typedef struct tagMSL_OPESDPPARAM { unsigned char proc_flg ; /**< Flag indicating processing status */ unsigned char reserved[3] ; /**< Boundary */ unsigned char cardid[16] ; /**< Card ID */ _MSL_OPESDPRSTRDATA data; /**< (1) Shared recovery information */ }_MSL_OPESDPPARAM ; Total size (1+3+16+44) = 64 bytes Shared recovery information (* Shared information of (1)) typedef union tagMSL_OPESDPRSTRDATA { _MSL_OPESDPRSTREXPO expo ; /**< (2) Export recovery information */ _MSL_OPESDPRSTRTCHG tch ; /**< (3) Title change recovery information */			

```

_MSL_OPESDPRSTRDEL del ;                /**< (4) PIM file deletion recovery
information */
_MSL_OPESDPRSTRDATA ;
Total size: Maximum sizes of (2), (3), and (4)(due to shared information) = 44 bytes

Export recovery information structure (* Structure of (2))
typedef struct tagMSL_OPESDPRSTREXPO
{
    unsigned char file_name[32] ;          /**< Name of the vObject generation file */
    unsigned short pimdi_exist ;           /**< Number of PIMDI's that exist */
    unsigned short pimdi_used ;            /**< Number of PIMDI's being used */
    unsigned short plink_srn ;              /**< Previous link SRN */
    unsigned short add_srn ;                /**< New SRN */
} _MSL_OPESDPRSTREXPO ;
Total size: (32+2+2+2+2) = 40 bytes

Title change recovery information structure (* Structure of (3))
typedef struct tagMSL_OPESDPRSTRTCHG
{
    unsigned short chg_srn ;                /**< SRN to be changed */
    unsigned char reserved[2] ;             /**< Boundary */
    unsigned char title[32] ;               /**< Name of the title before change */
} _MSL_OPESDPRSTRTCHG ;
Total size: (2+2+32) = 36 bytes

Export recovery information structure (* Structure of (4))
typedef struct tagMSL_OPESDPRSTRDEL
{
    unsigned char file_name[ 32 ] ;         /**< Name of the file to be deleted */
    unsigned short srn ;                     /**< SRN to be deleted */
    unsigned short plink_srn ;               /**< Previous link SRN */
    unsigned short nlink_srn ;              /**< Next link SRN */
    unsigned short pimdi_exist ;            /**< Number of PIMDI's that exist */
    unsigned short pimdi_used ;            /**< Number of PIMDI's being used */
    unsigned char reserved[2] ;             /**< Boundary */
} _MSL_OPESDPRSTRDEL ;
Total size: (32+2+2+2+2+2+2) = 44 bytes

```

11.77 Setting or Referencing Antenna Bar Level

Classification	Terminal setting		
Function name	Setting or referencing antenna bar level	Symbol	Msl_OpeAntennaIndSetRef
Function overview	<p>This function is used to set or reference the antenna level.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeAntennaIndSetRef(mode, level, data) ;		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
level	int	I	Specifies the antenna. MSL_ANTENNA_ECNO1: Antenna bar display Ec/No X1 MSL_ANTENNA_ECNO2: Antenna bar display Ec/No X2 MSL_ANTENNA_RSCP1: Antenna bar display RSCP Y1 MSL_ANTENNA_RSCP2: Antenna bar display RSCP Y2
data	unsigned char *	I/O	Data storage area
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			

11.78 Setting or Referencing Profiles

Classification	General function		
Function name	Setting or referencing profiles	Symbol	Msl_OpeProfileSetRef
Function overview	<p>This function is used to set or reference profiles.</p> <p>The following data items can be specified for profiles:</p> <p>Name: 32 bytes including first and last names</p> <p>Kana: 32 bytes including first and last names</p> <p>Telephone number: 3 numbers (26 columns per number)</p> <p>Telephone number icon: 3 icons</p> <p>Mail address: 3 addresses (50 bytes per address)</p> <p>Mail address icon: 3 icons</p> <p>Address and zip code: 7 bytes (for zip code) + 93 bytes (for address)</p> <p>Memo: 100 bytes</p> <p>Registered image: Flag indicating with or without (on or off)</p> <p>i-modeURL: 2 URLs</p> <p>i-modeURL icon: 2 icons (not used)</p>		
Include file	res_ope.h		
Calling sequence	void Msl_OpeProfileSetRef(mode, dat);		
Argument name	Type	I/O	Explanation
Mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
Dat	_MSL_PROFILE *	I/O	Profile data structure pointer
Return value	Type	I/O	Explanation
-	Void	-	-
Remarks	<ul style="list-style-type: none"> - Use a null for the end of the name, kana, telephone number, and mail address which are members of the profile structure. - Always register @ simultaneously with the mail address. Register nothing in Telephone number 1. - Determining whether images are registered or not can depend on whether information inside the image structure has information, 		

* set the initial value (zero clear) with no images.

[Profile data structure]

```
typedef struct tagMSL_PROFILE
```

```
{
    unsigned char    name_sei[MSL_KANJI_SEI_MAX+1];    /* Name
(last name)        */
    unsigned char    name_mei[MSL_KANJI_MEI_MAX+1];    /* Name
(first name)       */
    unsigned char    kana_sei[MSL_KANA_SEI_MAX+1];      /* Kana
(last name)        */
    unsigned char    kana_mei[MSL_KANA_MEI_MAX+1];      /* Kana
(first name)       */
    unsigned char    image;                            /* Flag indicating with
or without image (on or off) */
    _MSL_TELNO      dial[MSL_DIAL_NUM];                /*
Telephone number and icon */
    _MSL_MAILADR     mail_adr[MSL_MAILADR_NUM];         /* Mail
address and icon */
    _MSL_MEDADDRESS  address;                          /* Address
data */
    _MSL_MEDMEMO     memo;                             /* Memo data */
    _MSL_SIPADR      sip_adr[MSL_SIPADR_NUM];          /* Not used */
    _MSL_URL         imode_url[MSL_URL_NUM];           /* i-mode URL and
icon */
    _MSL_URL         wlan_url[MSL_WLANURL_NUM];        /* Not
used */
} _MSL_PROFILE ;
```

```
#define MSL_KANJI_SEI_MAX  32 /* Upper limit of kanji (last name) data
size */
```

```
#define MSL_KANJI_MEI_MAX  32 /* Upper limit of kanji (first name) data
size */
```

```
#define MSL_KANA_SEI_MAX   32 /* Upper limit of kana (last name) data
size */
```

```
#define MSL_KANA_MEI_MAX   32 /* Upper limit of kana (first
name) data size */
```

```
#define MSL_DIAL_NUM       3 /* Telephone number and
number of icons */
```

```
#define MSL_MAILADR_NUM    3 /* Mail address and number of icons */
```

```
#define MSL_URL_NUM        2 /* i-mode URL */
```

[Telephone number and icon structure]

```
typedef struct tagMSL_TELNO
```

```

{
    unsigned char    tel_no[MSL_DIALMAXB+1];    /*      Telephone
number          */
    unsigned short   tel_icon;                  /*      Telephone
number icon     */
} _MSL_TELNO ;

#define MSL_DIALMAXB    26    /*Size of dial data registered in the
telephone book within the mobile phone */

[Mail address and icon structure]
typedef struct tagMSL_MAILADR
{
    unsigned char    mail_adr[MSL_MAILMAX+1];    /*      Mail
address        */
    unsigned short   adr_icon;                    /*      Mail   address
icon          */
} _MSL_MAILADR ;

#define MSL_MAILMAX    50                /*      Maximum
number of mail addresses */

[Address data structure]
typedef struct tag_MSL_MEDADDRESS
{
    unsigned char    zipcode[MSL_ZIPCODEMAX+1]; /*      Zip
code          */
    unsigned char    data[MSL_ADDRESSMAX+1];    /*      Address
data          */
    unsigned short   icon;                        /* Icon information */
} _MSL_MEDADDRESS ;

#define MSL_ZIPCODEMAX    7                /* Maximum number of digits
for a zip code */
#define MSL_ADDRESSMAX    93                /* Maximum number of digits
for an address text */

[Memo data structure]
typedef struct tag_MSL_MEDMEMO
{
    unsigned char    data[MSL_MEMOMAX+1];        /*      Memo
data          */
    unsigned short   icon;                        /* Icon information */
}

```

```
} _MSL_MEDMEMO ;

#define MSL_MEMOMAX      100      /* Maximum number of digits
for a memo      */

[URL and icon structure]
typedef struct tagMSL_URL
{
    unsigned char    url[MSL_URL_MAX + 1]; /* URL      */
    unsigned short   url_icon;              /*      URL
icon      */
} _MSL_URL ;

#define MSL_URL_MAX      256      /* Upper limit of URL data
size
```

11.79 Registering or Referencing Serial Production Number

Classification	Serial processing		
Function name	Registering or referencing serial production number	Symbol	Msl_Ope_SerialNumSetRef
Function overview	<p>This function is called when D1010F (Registering serial production number) and D1030F (Referencing serial production number) is notified by an external terminal.</p> <ul style="list-style-type: none"> - When this function is called, it registers or references the serial production number according to the value specified in mode (second argument). - In registration processing, this function registers the serial production number notified by an external terminal in the two device information areas. - In reference processing, this function reads the serial production number from the two device information areas and sets the number in the APN-specified storage area. 		
Include file	res_ope.h		
Calling sequence	int Msl_Ope_SerialNumSetRef(mode , num);		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
bkl	int *	I/O	Serial production number storage area pointer In registration: Pointer of the area that stores the serial production number notified by an external terminal (8 bytes) In reference: Pointer of the area that stores the serial production number (8 bytes)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end (parameter error)
Remarks	<p>Serial production number: num[8];</p> <p>The data format to be passed to the resource management library is as shown below.</p> <p>* For example, when the data as shown below is notified by an external terminal, convert data of byte 3 or later and pass it to the resource management library.</p>		

When the serial production number is notified as the D1 command shown below for the data format in registration:

[0xD1][0x03][0x0F]

[0x00][0x01][0x02][0x03][0x04][0x05][0x06][0x07]

[0x08][0x09][0x0A][0x0B][0x0C][0x0D][0x0E][0x0F]

To obtain the data format to be notified of the resource management library, shift byte 4 to the left by 4 and perform OR with byte 5. Then, shift byte 6 to the left by 4 and perform OR with byte 7.

Use the above steps to convert data up to byte 19. Then, the following data format can be obtained:

[0x01][0x23][0x45][0x67][0x89][0xAB][0xCD][0xEF]

11.80 Referencing Nonvolatile Area (EEPROM) Initial Value

Classification	General function		
Function name	Referencing nonvolatile area initial value	Symbol	Msl_nonvolatile_init_read
Function overview	<p>This function is used to reference nonvolatile area (EEPROM) initial value data.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_nonvolatile_init_read (adr, size, data)		
Argument name	Type	I/O	Explanation
adr	unsigned short	I	Nonvolatile data address
size	unsigned short	I	Data size (1 to 256 bytes)
data	unsigned char *	I/O	Nonvolatile data storage pointer
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			

11.81 Referencing Nonvolatile Area (EEPROM)

Classification	General function		
Function name	Referencing nonvolatile area (EEPROM)	Symbol	Msl_nonvolatile_read
Function overview	<p>This function is used to reference nonvolatile area (EEPROM) data.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_nonvolatile_read (adr, size, data)		
Argument name	Type	I/O	Explanation
adr	unsigned short	I	Nonvolatile data address
size	unsigned short	I	Data size
data	unsigned char *	I/O	Nonvolatile data storage pointer
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			

11.82 Setting Nonvolatile Area (EEPROM)

Classification	General function		
Function name	Setting nonvolatile area (EEPROM)	Symbol	Msl_nonvolatile_write
Function overview	<p>This function is used to set nonvolatile area (EEPROM) data.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_nonvolatile_write(adr, size, data)		
Argument name	Type	I/O	Explanation
adr	unsigned short	I	Nonvolatile data addresss
size	unsigned short	I	Data size
data	unsigned char *	I/O	Nonvolatile data storage pointer
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks			

11.83 Setting or Referencing Priority Connection Destination Setting Data

Classification	General function		
Function name	Setting or referencing priority connection destination setting data	Symbol	Msl_OpePriorityDataSetRef
Function overview	<p>This function is used to set or reference the file system data regarding the priority connection destination setting data.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpePriorityDataSetRef(mode, no, data)		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies mode. MSL_FUNCSET: Setting processing MSL_FUNCREF: Reference processing
no	int	I	Priority connection destination setting data number MSL_PRIORITYALL: Data about all priority connection destinations (1 to 20) MSL_PRIORITY01: Priority connection destination setting 1 ~ MSL_PRIORITY20: Priority connection destination setting 20
data	unsigned short *	I/O	Priority connection destination setting data list (*1)
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks	<p>(*1)</p> <p>The upper-level application must allocate the following storage areas for the priority connection destination setting data lists:</p> <ul style="list-style-type: none"> - When data about all priority connection destinations (MSL_PRIORITYALL) is specified, => unsigned short * areas for 20 destinations - When priority connection destination setting 1 to 20 (MSL_PRIORITY01 to MSL_PRIORITY20) is specified, => Area for unsigned short 		

11.84 Recovery Request after Power-off

Classification	Service function		
Function name	Recovery request after power-off	Symbol	Msl_OpeRecoveryReq
Function overview	<p>This function provides the recovery function to be performed in the startup phase after power-off. This function checks the normality of the data file to be saved on JFFS2 (file system) or allocation directory.</p>		
Include file	res_ope.h		
Calling sequence	int Msl_OpeRecoveryReq (void) ;		
Argument name	Type	I/O	Explanation
void	-	-	No argument
Return value	Type	I/O	Explanation
ret	int	O	Processing result MSL_OK: Normal end MSL_NG: Abnormal end
Remarks	<p>This function assumes that it is started from the service manager at system start. This function performs only recovery processing regarding the following data:</p> <ul style="list-style-type: none"> - Recovery of user-defined reset processing - Compact desktop recovery - UIM certificate cache information recovery - Mailing list recovery - File system wrapper initialization 		

11.85 Collecting Display Color

Classification	Color pattern relation		
Function name	Collecting display color	Symbol	Res_DSP_ColorGet
Function overview	<p>This function is used to collect the display color for the specified display type from the color patterns.</p>		
Include file	res_dsp.h		
Calling sequence	int Res_DSP_ColorGet(no , type , color) ;		
Argument name	Type	I/O	Explanation
no	int	I	Color pattern number (*3) RES_DSP_CLR_CRNT Current color RES_DSP_CLR_PTN1 Standard RES_DSP_CLR_PTN2 Moon yellow RES_DSP_CLR_PTN3 Silent blue RES_DSP_CLR_PTN4 Twilight rose RES_DSP_CLR_PTN5 Midnight blue
type	int	I	Display type Specifies an identifier that indicates the display area and type.
color	_RES_DSP_COLOR *	O	Display color information structure pointer (*2)
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_DSP_OK: Normal end RES_DSP_NG: Abnormal end RES_DSP_PARAM_NG: Parameter error
Remarks			

*2 The calling source of this function must allocate an area for the following display color information structure.

*3 When the fixed display type ID is specified, the color pattern number is ignored.

[Display color information structure]

```
typedef struct {  
    unsigned short red;  
    unsigned short green;  
    unsigned short blue;  
} _RES_DSP_COLOR ;
```

11.86 Setting or Collecting Color Pattern Number

Classification	Color pattern relation		
Function name	Setting or collecting color pattern number	Symbol	Res_DSP_ColorPaletteSetRef
Function overview	<p>This function is used to change or reference the setting of the color pattern number currently specified.</p>		
Include file	res_dsp.h		
Calling sequence	int Res_DSP_ColorPaletteSetRef(mode , no) ;		
Argument name	Type	I/O	Explanation
mode	int	I	Specifies operation. RES_DSP_FUNCSET: Setting processing RES_DSP_FUNCREF: Reference processing
no	int	I	Color pattern number (*1)
Return value	Type	I/O	Explanation
ret	int	O	Processing result Color pattern number: Normal end (*1) RES_DSP_NG: Abnormal end RES_DSP_PARAM_NG: Parameter error
Remarks	<p>1) Setting processing</p> <ul style="list-style-type: none"> - noArgument is reflected only when RES_DSP_FUNCSET (Setting processing) is specified in mode. - When setting processing is completed normally, the pattern number of the current color pattern is specified for the return value. <p>2) Reference processing</p> <ul style="list-style-type: none"> - Even though a color pattern number is specified in noArgument, it is ignored in reference processing. <p>*1 Color pattern number</p> <pre> #define RES_DSP_CLR_PTN1 // Standard #define RES_DSP_CLR_PTN2 // Moon yellow #define RES_DSP_CLR_PTN3 // Silent blue #define RES_DSP_CLR_PTN4 // Twilight rose #define RES_DSP_CLR_PTN5 // Midnight blue </pre>		

11.87 Operating Text Memo

Classification	Text memo relation		
Function name	Operating text memo	Symbol	Res_TM_Memo_Ope
Function overview	<p>This function is used to register, correct, or reference the specified text memo.</p>		
Include file	res_tm.h		
Calling sequence	int Res_TM_Memo_Ope(mode, no, current_date, data) ;		
Argument name	Type	I/O	Explanation
Mode	Int	I	Specifies operation. RES_TM_FUNCREG: Registration processing RES_TM_FUNCMOD: Correction processing RES_TM_FUNCREF: Reference processing
No	Int	I	Text memo registration number: A value from 0 to RES_TM_FILE_MAX can be specified.
current_date	_RES_CAL_DATETIME *	I	Current time structure pointer (*1)
Data	_RES_TMDATA *	I/O	Text memo information (*2) Specifies an address of the text memo information storage area.
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_TM_OK: Normal end RES_TM_NG : Abnormal end RES_TM_PARAM_NG: Parameter error RES_TM_NO_ENTRY: Not registered RES_TM_NO_SPACE: Area shortage RES_TM_BUSY: Busy
Remarks			

- 1) Note
 - The calling source of this function must allocate an area for the address specified in dataArgument.
- 2) In registration processing
 - For area shortage, this function does not perform registration processing. Instead, it returns RES_TM_NO_SPACE (area shortage) as a return value.
 - When the specified text memo registration number has already been registered, the function returns RES_TM_BUSY (busy).
 - When data is greater than the maximum data size, data up to the maximum size is registered.
 - The creation date and last update date are updated using this function at registration or update based on the specified current time.
- 3) In correction processing
 - When the specified text memo registration number is not registered, this function returns RES_TM_NO_ENTRY (not registered).
 - For area shortage, this function does not perform registration processing. Instead, it returns RES_TM_NO_SPACE (area shortage) as a return value.
 - When data is greater than the maximum data size, data up to the maximum size is registered.
 - The creation date and last update date are updated using this function at registration or update based on the specified current time.
- 4) In reference processing
 - When the specified text memo registration number is not registered, this function returns RES_TM_NO_ENTRY (not registered).

*1 Structures of data to be set

[Current time structure]

typedef struct tagRES_CAL_DATETIME

```
{
    unsigned short Year; /* Year: 2001 to 2099 */
    unsigned char Mon; /* Month: 1 to 12 */
    unsigned char Day; /* Day: 1 to 31 */
    unsigned char Week; /* Day of the week: 0 (Sun.) to 6 (Sat.) */
    unsigned char Hour; /* Hour: 0 to 23 */
    unsigned char Min; /* Minute: 0 to 59 */
    unsigned char Sec; /* Second: 0 to 59 */
} _RES_CAL_DATETIME ;
```

*2

[Text memo information structure]

typedef struct tag_RES_TMDATA

```
{
    _RES_CAL_DATETIME Dcreate; /* Creation date and
time */
```

```

    _RES_CAL_DATETIME Modified;      /* Last update date and
time */
    unsigned char Category;          /* Classification (*3) */
    unsigned char TmMemoData[RES_TM_TMMAX + 1]; /* Area for reading one
text memo */
} _RES_TMDATA ;

#define RES_TM_TMMAX 512 /* Maximum text memo registration
buffer */

*3
/* Classification list */
#define RES_TM_DEFAULT /* No (default) */
#define RES_TM_PERSONAL /* Private */
#define RES_TM_HOLIDAY /* Holiday */
#define RES_TM_TRAVEL /* Travel */
#define RES_TM_BUSINESS /* Business */
#define RES_TM_MEETING /* Meeting */

```

11.88 Deleting Text Memo

Classification	Text memo relation		
Function name	Deleting text memo	Symbol	Res_TM_Memo_Del
Function overview	<p>This function is used to delete a specified text memo data or all text memo data items.</p>		
Include file	res_tm.h		
Calling sequence	int Res_TM_Memo_Del(mode , no) ;		
Argument name	Type	I/O	Explanation
mode	Int	I	Operation type RES_TM_FUNCDEL: Deletes one data specified. RES_TM_FUNCDEL_ALL: Deletes all data items.
no	Int	I	Text memo registration number A value from 0 to RES_TM_FILE_MAX can be specified.
Return value	Type	I/O	Explanation
ret	Int	O	Processing result: RES_TM_OK: Normal end RES_TM_NG: Abnormal end RES_TM_PARAM_NG: Parameter error RES_TM_NO_ENTRY: Not registered
Remarks	<p>1) Deleting specified data</p> <ul style="list-style-type: none"> - When the specified text memo registration number is not registered in deleting the specified data, this function returns RES_TM_NO_ENTRY (not registered). - no Argument is reflected only for deleting the specified data. <p>2) Deleting all items</p> <ul style="list-style-type: none"> - When no text memo is registered in deleting all data items, this function returns RES_TM_OK. - When no data was deleted in deleting all data items, this function returns RES_NG. 		

11.89 Referencing Text Memo List

Classification	Text memo relation		
Function name	Referencing text memo list	Symbol	Res_TM_Memo_List
Function overview	<p>This function is used to collect the specified text memo data for list display.</p>		
Include file	res_tm.h		
Calling sequence	int Res_TM_Memo_List(no , data_size, data) ;		
Argument name	Type	I/O	Explanation
No	int	I	Text memo registration number A value from 0 to RES_TM_FILE_MAX can be specified.
data_size	int	I	Size for reading text memo data A value from 1 to RES_TM_TMMAX can be specified.
Data	unsigned char *	O	Text memo data Specifies an address of the text memo data storage area.
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_TM_OK: Normal end RES_TM_NG: Abnormal end RES_TM_PARAM_NG: Parameter error RES_TM_NO_ENTRY: Not registered
Remarks	<pre>#define RES_TM_TMMAX 512 /* Maximum text memo registration buffer */ #define RES_TM_FILE_MAX 9 /* Maximum text memo registration number */</pre> <p>Notes</p> <ul style="list-style-type: none"> - The calling source of this function must allocate an area of the address to be specified in dataArgument. - Allocate an area of data_size + 1 or greater for the area to be specified in dataArgument. 		

11.90 Collecting Number of Text Memo Data Items

Classification	Text memo relation		
Function name	Collecting number of text memo data items	Symbol	Res_TM_Memo_Cnt
Function overview	<p>This function is used to collect the number of text memos registered.</p>		
Include file	res_tm.h		
Calling sequence	int Res_TM_Memo_Cnt(void) ;		
Argument name	Type	I/O	Explanation
-	void	-	-
Return value	Type	I/O	Explanation
cnt	int	O	Processing result Number of text memos registered: Normal end (*1) RES_TM_NG: Abnormal end
Remarks	<p>*1 When no text memo is registered, this function returns 0 as a return value.</p>		

11.91 Text Memo Operation

Classification	Text memo relation		
Function name	Registering text memo	Symbol	Res_TM_Memo_Write
Function overview	<p>This function is used to register a text memo of the specified number.</p>		
Include file	res_tm.h		
Calling sequence	int Res_TM_Memo_Write(no, data) ;		
Argument name	Type	I/O	Explanation
no	int	I	Text memo registration number A value from 0 to RES_TM_FILE_MAX can be specified.
data	_RES_TMDATA *	I/O	Text memo information (*1) Specifies an address of the text memo information storage area.
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_TM_OK: Normal end RES_TM_NG: Abnormal end RES_TM_PARAM_NG: Parameter error RES_TM_NO_SPACE: Area shortage RES_TM_BUSY: Busy
Remarks			

Notes:

- For area shortage, this function does not perform registration processing. Instead, it returns RES_TM_NO_SPACE (area shortage) as a return value
- When the specified text memo registration number has already been registered, the function returns RES_TM_BUSY (busy).
- When data is greater than the maximum data size, data up to the maximum size is registered.

*1

[Text memo information structure]

typedef struct tag_RES_TMDATA

```
{
    _RES_CAL_DATETIME Dcreate;          /* Creation date and time          *2
*/
    _RES_CAL_DATETIME Modified;         /* Last update date and time      *2
*/
    unsigned char Category;             /* Classification                  *3 */
    unsigned char TmMemoData[RES_TM_TMMAX + 1]; /* Area for reading one
text memo */
} _RES_TMDATA ;
```

```
#define RES_TM_TMMAX      512          /* Maximum text memo registration
buffer */
```

*2

[Current time structure]

typedef struct tagRES_CAL_DATETIME

```
{
    unsigned short Year;    /* Year: 2001 to 2099    */
    unsigned char Mon;      /* Month: 1 to 12        */
    unsigned char Day;      /* Day: 1 to 31          */
    unsigned char Week;     /* Day of the week: 0 (Sun.) to 6 (Sat.) */
    unsigned char Hour;     /* Hour: 0 to 23         */
    unsigned char Min;      /* Minute: 0 to 59       */
    unsigned char Sec;      /* Second: 0 to 59       */
} _RES_CAL_DATETIME ;
```

```
*3
/* Classification list */
#define RES_TM_DEFAULT          /* No (default) */
#define RES_TM_PERSONAL        /* Private      */
#define RES_TM_HOLIDAY         /* Holiday    */
#define RES_TM_TRAVEL           /* Travel     */
#define RES_TM_BUSINESS         /* Business   */
#define RES_TM_MEETING         /* Meeting    */
```

11.92 Wording Collection Processing

Classification	Wording data supporting function		
Function name	Wording collection processing	Symbol	Res_WD_LanguageRd
Function overview	<p>This function is used to collect a character string in the specified language.</p>		
Include file	res_wd.h		
Calling sequence	int Res_WD_LanguageRd(mode, str_id, data) ;		
Argument name	Type	I/O	Explanation
Mode	int	I	Collection language RES_WD_JAPANESE: Japanese RES_WD_ENGLISH: English
str_id	int	I	Collection ID (*1)
Data	char *	O	Collection character string (*2)
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_WD_OK: Normal end RES_WD_NG: Abnormal end (parameter error)
Remarks	<p>*2 Allocate an area having the size that can output data for the collection character string storage area. Specify ¥0 at the end of the character string to be output to the collection character string storage area using this function.</p> <p><u>Note:</u> <u>If the storage area of the specified collection character string is smaller than the collection character string to be output, an area destruction may occur.</u></p>		

11.93 Wording Size Collection Processing

Classification	Wording data supporting function		
Function name	Wording size collection processing	Symbol	Res_WD_LangSizeGet
Function overview	<p>This function is used to collect the size of the specified collection character string (collection language or ID).</p>		
Include file	res_wd.h		
Calling sequence	int Res_WD_LangSizeGet(mode, str_id, data_size);		
Argument name	Type	I/O	Explanation
mode	int	I	Collection language RES_WD_JAPANESE: Japanese RES_WD_ENGLISH: English
str_id	int	I	Collection ID (*1)
data_size	unsigned int *	O	Collection character string size
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_WD_OK: Normal end RES_WD_NG: Abnormal end (parameter error)
Remarks			

11.94 Request to Collect Image Display Data

Classification	Image data supporting function		
Function name	Request to collect image display data	Symbol	Res_IMG_DataRead
Function overview	<p>This function is used to collect image display data registered.</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_DataRead(img_no , img_info) ;		
Argument name	Type	I/O	Explanation
img_no	unsigned short	I	Image display data save number (*1) RES_IMG_NUMBER_MIN ~ RES_IMG_NUMBER_MAX : Image folder image RES_IMG_WAIT_IMOTION : Standby video image RES_IMG_PROFILE : Profile image
img_info	_RES_IMG_INFO *	I/O	Image display data structure (see Remarks.)
Return value	Type	I/O	Explanation
Ret	int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error RES_IMG_DATANOT: Data not registered RES_IMG_WRITE_EXE: Data being written RES_IMG_READ_NG: Reading failed.
Remarks	<pre>[Image display data structure] typedef struct tagRES_IMG_INFO { _RES_IMG_HEADINFO head ; /* [I/O] Image display data header */ unsigned char *data ; /* [I/O] Image data</pre>		


```

pointer          (*2) */
} _RES_IMG_INFO ;

[Image display data header structure]
typedef struct tagRES_IMG_HEADINFO
{
    unsigned short   type ; /* [-O] Image data type          (*3) */
    unsigned long    length ; /* [I/O] Image data size (unit: bytes) (*4) */
                                /* IN: Image data area */
                                /* Size allocated by data of _RES_IMG_INFO */
    /*
                                /* OUT: Size after collecting image
data */
    unsigned char    title[RES_IMG_TITLE_MAX+1] ; /* [-O] Storage area for image
data title character string */
    unsigned char    cp_flg ; /* [-O] File restriction          (*5) */
    unsigned char    edit_flg ; /* [-O] Image processing
flag (*6) */
    unsigned char    sv_flg ; /* (Not used) (*7) */
    unsigned char    fname[RES_IMG_FNAME_MAX+1] ; /* [-O] Storage area for
file name character string */
    unsigned char    extension[RES_IMG_EXTENSION_MAX+1] ;
                                /* [-O] Storage area for extension character
string (*8) */
    unsigned short   index ; /* [-O] Image display data save
number (*9) */
    unsigned long    counter ; /* [-O] Image registration
number (*10) */
    unsigned char    sv_data[RES_IMG_DATE_MAX] ; /* [-O] Save date and
time (*11) */
    int              seq_no ; /* [-O] Sequential number (*16) */
    /*
    _RES_IMG_DRMINFO drm_info ; /* [-O] DRM
information */
    _RES_IMG_SWFINFO swf_info ; /* [-O] SWF attribute
information */
                                /* * This value is defined only for saving vector
graphics. */
} _RES_IMG_HEADINFO ;
#define RES_IMG_TITLE_MAX (18) /* Image data title character
string size */
#define RES_IMG_FNAME_MAX (36) /* File name character string
size */
#define RES_IMG_EXTENSION_MAX (4) /* Extension name character
string size */
#define RES_IMG_DATE_MAX (5) /* Save date and time storage

```

```

area
    */

[DRM information structure]
typedef struct tagRES_IMG_DRMINFO
{
    unsigned char    sv_mode ;           /*          [-/O]          Save
information          (*12) */
    unsigned char    uim_flg ;           /*          [-/O]          UIM    ID    existence
flag          (*13) */
    unsigned char    uim_no[RES_IMG_UIMNO_MAX] ; /* [-/O] UIM production
number          */
} _RES_IMG_DRMINFO ;

#define    RES_IMG_UIMNO_MAX            (10)          /* UIM    ID    information
size          */

[SW attribute information structure] * Use this structure only for saving vector graphics.
typedef struct tagRES_IMG_SWFINFO
{
    unsigned char    play ;              /* [-/O] Replay method          (*14) */
    unsigned short   width ;             /* [-/O] width attribute (landscape
size)          */
    unsigned short   height ;            /* [-/O] height attribute (portrait
size)          */
    unsigned short   quality ;           /* [-/O] quality attribute (file image
quality)      (*15) */
    int              bgcolor ;           /* [-/O] bgcolor attribute (player area
background color) */
} _RES_IMG_SWFINFO ;

(*1) For the image display data save number, use E-5, [Request to Collect List of
Image Display Data Headers] to collect the target image display data header
information. Then, use the image display data save number (index) stored in the
information.

(*2) Before calling this function, allocate an area having the size of the image display
data structure (_RES_IMG_INFO).

(*3) Image data type setting values (used at registration)
    RES_IMG_TYPE_GIF:  Gif data
    RES_IMG_TYPE_GIF87A:  Gif87a data
    RES_IMG_TYPE_GIF89A:  Gif89a data
    RES_IMG_TYPE_JPEG:  Jpeg data
    RES_IMG_TYPE_SWF:  SWF (vector graphics) data
    RES_IMG_TYPE_IFM:  IFM (download frame) data

```

For GIF, RES_IMG_TYPE_GIF (standard GIF format) or RES_IMG_TYPE_GIF87A/RES_IMG_TYPE_GIF89A (more detailed) can be specified optionally.

(*4) When the result is returned, this value is replaced with the length value of the image display data actually read.

(*5) File restriction setting values (used at registration)

RES_IMG_CP_ON: With file restriction

RES_IMG_CP_OFF: Without file restriction

(*6) Image processing flag setting values (used at registration)

RES_IMG_EDIT_YES: With image processed

RES_IMG_EDIT_NO: Without image processed

(*7) External save permission flag setting value (not used)

RES_IMG_SV_0: Without identifier

RES_IMG_SV_1: Free from copyrights

RES_IMG_SV_2: Data encoding only

RES_IMG_SV_3: User bind

RES_IMG_SV_4: Set bind

RES_IMG_SV_5: External save disabled

(*8) Effective extensions are “.jpg”, “.gif”, “.swf”, and “.ifm” (used at registration).

(*9) Image display data save number setting values (used at registration)

RES_IMG_NUMBER_MIN to RES_IMG_NUMBER_MAX: Image folder image

RES_IMG_WAIT_IMOTION: Standby video image

RES_IMG_PROFILE: Profile image

(*10) Sorting the date order depends on the save date and time of the image display data header structure. At overwrite registration, if updating the save date and the counter member are cleared to 0, the order sorted using the date is changed. At overwrite registration, if both items are not changed, the order is not changed. (This member is used at overwrite registration.)

(*11) Specify the save date and time as follows (used at registration):

sv_data[0]:Year, [1]:Month, [2]:Day, [3]:Hour, [4]:Minute

=> Use a number between 00 and 99 (decimal) to specify the above values.

Set 0xff.

* When this member is not used (e.g., mobile phone date and time not set), specify 0xff for each of the above items.

* Set the last two digits of the year for the year information.

* The resource management does not check the validity of the save date and time.

(Example: 03/02/29, 0:60)

- (*12) Save source information setting values (used at registration)
 - RES_IMG_SVMODE_MAIL: Registration from the mail attachment.
 - RES_IMG_SVMODE_DL: Registration from download
 - RES_IMG_SVMODE_CAM: Camera registration (shooting or process)
 - RES_IMG_SVMODE_EXT_MEMORY: Registration from the external memory
 - RES_IMG_SVMODE_TV_TELEPHONE: Registration from the TV phone

- (*13) UIM ID existence flag setting values (used at registration)
 - RES_IMG_UIMID_NONE: With UIM ID
 - RES_IMG_UIMID_EXIST: Without UIM ID

- (*14) Replay method setting values (used at registration)
 - RES_IMG_SWF_PLAY_INLINE: Inline replay
 - RES_IMG_SWF_PLAY_INTERACTIVE: Interactive replay

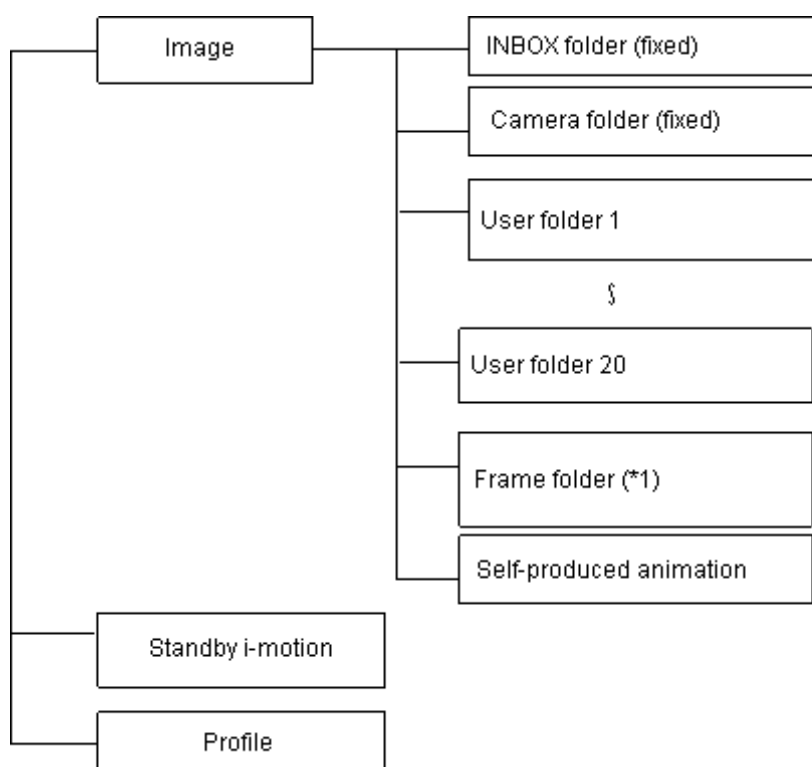
- (*15) quality attribute setting values (used at registration)
 - RES_IMG_SWF_QUALITY_HIGH: high
 - RES_IMG_SWF_QUALITY_MEDIUM: medium
 - RES_IMG_SWF_QUALITY_LOW: low

- (*16) Sequential number (used at registration)
 - New registration: The upper-level application need not set a sequential number because the resource library set it.
 - Overwrite registration: Set an image sequential number already registered.
 - * Unlike image numbers, sequential numbers are not reused. Therefore, they can be identified because “sequential number + 1” is always set for a new image. However, a sequential number is intType. If sequential numbers exceed the maximum int value, the new sequential number starts with 1.
 - * Standby video and profile images have sequential number 0 (fixed).

- * Supplementary information
 - #define RES_IMG_GET_INFO_MAX (400) /* Total number of image display data items */
 - #define RES_IMG_DATA_MAX_UXGA (614400) /* Maximum image data size (UXGA): bytes */
 - #define RES_IMG_DATA_MAX_VGA (102400) /* Maximum image data size (VGA): bytes */
 - #define RES_IMG_DATA_MAX_S (20480) /* Maximum image data size (MINI): bytes */

- The figure below shows the image display data folder configuration that the resource management (image data supporting function) uses.

Note that the folder names described in this manual comply with those in the following figure:



(*1) Only download frame images can be used.

Number		Size	
Maximum	Minimum	Maximum size per frame (KB)	Minimum size per frame (KB)
400	3	600	5

11.95 Request to Register Image Display Data

Classification	Image data supporting function		
Function name	Request to register image display data	Symbol	Res_IMG_DataSave
Function overview	<p>This function is used to register or overwrite image data.</p> <p>This function allows the user to register GIF, JPEG, SWF, and IFM files.</p> <p>Specify the SWF attribute information of the image data header structure to save vector graphics. Specify 0 in each member if the information is not used.</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_DataSave(img_no , img_info) ;		
Argument name	Type	I/O	Explanation
img_no	unsigned short	I	<p>At new registration: Save destination folder number (*1)</p> <p>RES_IMG_FOLDER_INBOX: INBOX folder</p> <p>RES_IMG_FOLDER_CAM: Camera folder</p> <p>RES_IMG_FOLDER_USER_01: User folder 1</p> <p>~ ~</p> <p>RES_IMG_FOLDER_USER_20: User folder 20</p> <p>RES_IMG_FOLDER_FRAME: Frame folder</p> <p>RES_IMG_FOLDER_WAIT_IMOTION: Standby video</p> <p>* Because a profile image is always overwritten, register it with the following specifications at overwrite registration:</p> <p>At overwrite registration: Image display data save number</p> <p>RES_IMG_NUMBER_MIN to RES_IMG_NUMBER_MAX : Image folder image</p>

			RES_IMG_WAIT_IMOTION : Standby video image RES_IMG_PROFILE : Profile image
img_info	_RES_IMG_INFO *	I/O	Image display data structure (see Remarks)
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error RES_IMG_WRITE_NG: Writing data failed. RES_IMG_NO_SPACE: Free space shortage
Remarks	<pre> [Image display data structure] typedef struct tagRES_IMG_INFO { _RES_IMG_HEADINFO head ; /* [I/O] Image display data header */ unsigned char *data ; /* [I/-] Image data pointer (*2) */ } _RES_IMG_INFO ; [Image display data header structure] typedef struct tagRES_IMG_HEADINFO { unsigned short type ;/* [I/-] Image data type */ unsigned long length ;/* [I/-] Image data size (unit: bytes) */ unsigned char title[RES_IMG_TITLE_MAX+1] ; /* [I/-] Storage area for image data title character string (note) */ unsigned char cp_flg ;/* [I/-] File restriction (*3) */ unsigned char edit_flg ;/* [I/-] Image processing flag */ unsigned char sv_flg ; /* (Not used) */ unsigned char fname[RES_IMG_FNAME_MAX+1] ; /* [I/-] Storage area for file name character string (note) */ unsigned char extension[RES_IMG_EXTENSION_MAX+1] ; /* [I/-] Storage area for extension character string (note) */ unsigned short index ; /* [-/-] At overwrite registration: (*4) */ /* Image display data save number */ /* [-/O] At new registration: Image display data save number */ </pre>		


```

    unsigned long    counter ;                /* [I/O] At overwrite registration: Image
registrationnumber    (*5) */
                                /* [-/O] At new registration: Image registration
number                */
    unsigned char    sv_data[RES_IMG_DATE_MAX] ; /* [I/-] Save date and
time                */
    int              seq_no ;                /* [I/O] Sequential
number                (*6) */
    _RES_IMG_DRMINFO drm_info ;                /* [I/-] DRM
information            */
    _RES_IMG_SWFINFO swf_info ;                /* [I/-] SWF attribute
information            */
                                /* * This value is reflected only for saving vector
graphics. */
} _RES_IMG_HEADINFO ;

```

[DRM information structure]

```
typedef struct tagRES_IMG_DRMINFO
```

```

{
    unsigned char    sv_mode ;                /* [I/-] Save source information
    unsigned char    uim_flg ;                /* [I/-] UIM ID existence flag
    unsigned char    uim_no[RES_IMG_UIMNO_MAX] ; /* [I/-] UIM production
number                */
} _RES_IMG_DRMINFO ;

```

[SWF attribute information structure] * Use this structure only for saving vector graphics.

```
typedef struct tagRES_IMG_SWFINFO
```

```

{
    unsigned char    play ;                /* [I/-] Replay method
    unsigned short    width ;                /* [I/-] width attribute (landscape size)
    unsigned short    height ;                /* [I/-] height attribute (portrait size)
    unsigned short    quality ;                /* [I/-] quality attribute (file image quality)
    int              bgcolor ;                /* [I/-] bgcolor attribute (player area background
color)    */
} _RES_IMG_SWFINFO ;

```

* See “Request to Collect Image Display Data,” for details of the values set in each structure member. If a value other than the specified one is set in the specified member, a parameter error (RES_PARAM_NG) is returned.

(*1) Convert the “save destination folder number” to be set at new registration to the (image display data save number) using the OUT information to return it.
At overwrite registration, use the converted value (image display data save number: index).

(*2) The application must allocate an area for storing image display data and set a pointer in data (_RES_IMG_INFO structure member). Set the image data size in length of the _RES_IMG_INFO structure.

* If the image display data to be registered is greater than 600 KB, a parameter error (RES_PARAM_NG) is returned.

(*3) This flag indicates file restriction information, and does not indicate whether an external output (mail attachment, etc.) is enabled or not.

(*4) At overwrite registration: Information => The image display data save number need not be specified because the value set in `img_no` is specified.

At new registration: OUT information => image display data save number

* index manages the following images:

(1) Camera shooting image (2) Processing image (3) Download image (4) Mail-attached image (6) TV phone image (7) External memory image (8) Frame (download frame) image

* One standby video image and one profile image can be used. Therefore, they are fixed values (define values at overwrite registration).

(*5) At overwrite registration: IN information => Image registration number

* To change a display order sorted using the date, clear this member to 0.

OUT information => Image registration number after update

At new registration: Image registration number

(*6) Sequential number

New registration: The upper-level application need not set a sequential number because the resource library set it.

Overwrite registration: Set an image sequential number already registered.

* Unlike image numbers, sequential numbers are not reused. Therefore, they can be identified because (sequential number + 1) is always set for a new image.

However, a sequential number is `intType`. If sequential numbers exceed the maximum `int` value, the new sequential number starts with 1.

* Standby video and profile images have sequential number 0 (fixed).

Note: Specify a null at the end of a character string. The resource management service does not check the character string to be set in the member (disabled character check, etc.), but registers the character string. Each calling source must check the character strings.

11.96 Request to Erase Image Display Data

Classification	Image data supporting function		
Function name	Request to erase image display data	Symbol	Res_IMG_DataErase
Function overview	<p>This function is used to erase the specified image display data.</p> <p>When a folder is specified, all image display data items in the folder are erased.</p> <p>This function does not erase folders. (To erase folders, see “Request to Erase Image Display Data Folder.”)</p> <p>This function supports only the download frames.</p> <p>* Preinstall (frame) images are not supported.</p>		

Include file	res_img.h		
Calling sequence	int Res_IMG_DataErase(img_no) ;		
Argument name	Type	I/O	Explanation
img_no	unsigned short	I	<p><Specifying one data item> Image display data save number (*1) RES_IMG_NUMBER_MIN to RES_IMG_NUMBER_MAX : Image folder image RES_IMG_WAIT_IMOTION : Standby video image RES_IMG_PROFILE : Profile image</p> <p><Specifying deletion of all data items (deleting a folder)> Folder number to be deleted (*2) RES_IMG_FOLDER_ALL: All image folders RES_IMG_FOLDER_INBOX: INBOX folder RES_IMG_FOLDER_CAM : Camera folder RES_IMG_FOLDER_USER_01: User folder 1 ~ ~ RES_IMG_FOLDER_USER_20: User folder 20 RES_IMG_FOLDER_FRAME: Frame folder RES_IMG_FOLDER_WAIT_IMOTION: Standby video RES_IMG_FOLDER_PROFILE: Profile image</p> <p>* When RES_IMG_FOLDER_ALL is specified, all folders excluding profile and standby video images are erased.</p>
Return value	Type	I/O	Explanation

ret	int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error RES_IMG_DATANOT: Data not registered
Remarks	<p>(*1) For the image display data save number, use E-5, "Request to Collect List of Image Display Data Headers" to collect the target image display data header information. Then, use the image display data save number (index) stored in the information.</p> <p>(*2) When RES_IMG_FOLDER_ALL is specified, the target folders are as follows:</p> <ul style="list-style-type: none"> - INBOX folder, camera folder, user folders 1 to 20, frame folder 		

11.97 Request to Write Image Display Data Title

Classification	Image data supporting function		
Function name	Request to write image display data title	Symbol	Res_IMG_TitleWrite
Function overview	<p>This function is used to write a title of the image display data.</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_TitleWrite(img_no , img_titlestr) ;		
Argument name	Type	I/O	Explanation
img_no	unsigned short	I	Image display data save number (*1) RES_IMG_NUMBER_MIN to RES_IMG_NUMBER_MAX
img_titlestr	unsigned char *	I	Pointer of the image display data title character string (*2)
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error RES_IMG_DATANOT: Data not registered
Remarks	<p>(*1) For the image display data save number, use E-5, "Request to Collect List of Image Display Data Headers" to collect the target image display data header information. Then, use the image display data save number (index) stored in the information.</p> <p>(*2) Specify a null at the end of a character string.</p>		

11.98 Request to Collect List of Image Display Data Headers

Classification	Image data supporting function		
Function name	Request to collect list of image display data headers	Symbol	Res_IMG_TitleRead_List
Function overview	<p>This function is used to collect image display data header information for the specified number. * Information can be sorted and collected according to the condition specification.</p> <p>This function supports only the download frames.</p> <p>* Preinstall (frame) images are not supported.</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_TitleRead_List(folder_no , sortkey , start_no , get_cnt , headinfo);		
Argument name	Type	I/O	Explanation
folder_no	unsigned short	I	Folder number (*1) RES_IMG_FOLDER_ALL: All image folders RES_IMG_FOLDER_INBOX: INBOX folder RES_IMG_FOLDER_CAM: Camera folder RES_IMG_FOLDER_USER_01: User folder 1 ~ ~ RES_IMG_FOLDER_USER_20: User folder 20 RES_IMG_FOLDER_FRAME: Frame folder RES_IMG_FOLDER_WAIT_IMOTION: Standby video RES_IMG_FOLDER_PROFILE: Profile image
sortkey	unsigned short	I	Sort key RES_IMG_TIME_ASC: File date, starting with oldest RES_IMG_TIME_DESC: File data, starting with newest RES_IMG_TITLE_ASC: Title name, ascending order RES_IMG_TITLE_DESC: Title name,

			descending order RES_IMG_SIZE_ASC: Image display data size, starting with smallest RES_IMG_SIZE_DESC: Image display data size, starting with greatest RES_IMG_GET_METHOD: Order of file collection
start_no	unsigned short	I	Collection start position (1 to RES_IMG_GET_INFO_MAX(400)) (*2)
get_cnt	unsigned short	I	Number of collection data items (1 to RES_IMG_GET_INFO_MAX(400)) (*3)
headinfo	_RES_IMG_HEADINFO *	O	Pointer of image display data header information structure list (*4)
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error RES_IMG_DATANOT: Data not registered
Remarks	<p>(*1) When RES_IMG_FOLDER_ALL is specified, the target folders are as follows:</p> <ul style="list-style-type: none"> - INBOX folder, camera folder, user folders 1 to 20, frame folder <p>(*2) Specify the start position of the image display data header information to be collected.</p> <p>* When RES_IMG_FOLDER_ALL is specified in folder_no, this member becomes invalid.</p> <p>(*3) Specify the number of image display data header information collected.</p> <p>* To collect all image display data items in the folder, specify RES_IMG_GET_INFO_MAX(400).</p> <p>* When RES_IMG_FOLDER_ALL is specified in folder_no, this member becomes invalid.</p> <p>* When the collection start position and the range of the number of data items collected are incorrect, a parameter error (RES_IMG_PARAM_NG) occurs.</p> <p>Example: Collection start position: 100 + number of data items collected: 400, the system assumes NG because out of the range is specified.</p> <p>(*4) Allocate an area for the size specified in get_cnt for the pointer of image display data header information structure list. (Image display data header information structure x area for the number of data items collected)</p>		

```

[Image display data header structure]
typedef struct tagRES_IMG_HEADINFO
{
    unsigned short    type ;                /* [-/O] Image data type                */
    unsigned long     length ;              /* [-/O] Image data size (unit: bytes)   */
    unsigned char     title[RES_IMG_TITLE_MAX+1] ; /* [-/O] Storage area for image data
title character string */
    unsigned char     cp_flg ;              /* [-/O] File restriction                */
    unsigned char     edit_flg ;            /* [-/O] Image processing flag           */
    unsigned char     sv_flg ;              /* (Not used)                            */
    unsigned char     fname[RES_IMG_FNAME_MAX+1] ; /* [-/O] Storage area for file
name character string */
    unsigned char     extension[RES_IMG_EXTENSION_MAX+1] ;
/* [-/O] Storage area for extension character
string */
    unsigned short    index ;              /* [-/O] Image display data save
number */
    unsigned long     counter ;             /* [-/O] Image registration
number */
    unsigned char     sv_data[RES_IMG_DATE_MAX] ; /* [-/O] Save date and
time */
    int               seq_no ;              /* [-/O] Sequential number               */
    _RES_IMG_DRMINFO  drm_info ;           /* [-/O] DRM
information */
    _RES_IMG_SWFINFO  swf_info ;           /* [-/O] SWF attribute
information */
/* * This value is defined only for saving vector
graphics */
} _RES_IMG_HEADINFO ;

[DRM information structure]
typedef struct tagRES_IMG_DRMINFO
{
    unsigned char     sv_mode ;             /* [-/O] Save source
information */
    unsigned char     uim_flg ;             /* [-/O] UIM ID existence flag           */
    unsigned char     uim_no[RES_IMG_UIMNO_MAX] ; /* [-/O] UIM production
number */
} _RES_IMG_DRMINFO ;

[SWF attribute information structure] * Use this structure only for saving vector graphics
typedef struct tagRES_IMG_SWFINFO
{
    unsigned char     play ;               /* [-/O] Replay method                   */

```



```

        unsigned short width ;           /* [-/O] width attribute (landscape
size) */
        unsigned short height ;         /* [-/O] height attribute (portrait
size) */
        unsigned short quality ;        /* [-/O] quality attribute (file image
quality) */
        int bgcolor ;                   /* [-/O] bgcolor attribute (player area background
color) */
} _RES_IMG_SWFINFO ;

```

* See "Request to Collect Image Display Data," for details of the values set in each structure member.

11.99 Request to Collect Image Display Data Header Information

Classification	Image data supporting function		
Function name	Request to collect image display data header information	Symbol	Res_IMG_TitleRead
Function overview	<p>This function is used to specify the image display data save number and collect the specified image display data header information.</p> <p>When an image save number whose image display data is not registered is specified, return value RES_IMG_DATANOT (data not registered) is returned.</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_TitleRead(img_no , headinfo) ;		
Argument name	Type	I/O	Explanation
img_no	unsigned short	I	Image display data save number (*1) RES_IMG_NUMBER_MIN to RES_IMG_NUMBER_MAX : Image folder image RES_IMG_WAIT_IMOTION : Standby video image RES_IMG_PROFILE : Profile image
headinfo	_RES_IMG_HEADINFO *	O	Pointer of image display data header information structure list (*2)
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error RES_IMG_DATANOT: Data not registered
Remarks			

(*1) For the image display data save number, use E-5, "Request to Collect List of Image Display Data Headers" to collect the target image display data header information. Then, use the image display data save number (index) stored in the information.

(*2) Allocate an area for one data item for the pointer of image display data header information structure list (headinfo).

* See "Request to Collect List of Image Display Data Headers" for details of the structure member.

11.100 Request to Rename Image Display Data File

Classification	Image data supporting function		
Function name	Request to rename image display data file	Symbol	Res_IMG_FilenameWrite
Function overview	<p>This function is used to rename an image display data file.</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_FilenameWrite(img_no , fnamestr);		
Argument name	Type	I/O	Explanation
img_no	unsigned short	I	Image display data save number (*1) RES_IMG_NUMBER_MIN to RES_IMG_NUMBER_MAX
fnamestr	unsigned char *	I	Pointer of image display data file name character string (*2)
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error RES_IMG_DATANOT: Data not registered
Remarks	<p>(*1) For the image display data save number, use E-5, "Request to Collect List of Image Display Data Headers" to collect the target image display data header information. Then, use the image display data save number (index) stored in the information.</p> <p>(*2) The maximum size of a character string is: file name (RES_IMG_FNAME_MAX: 36) + extension (RES_IMG_EXTENSION_MAX: 4). Specify a null at the end of the character string. * The extension must include a dot (.).</p>		

11.101 Request to Collect Number of Image Display Data Items Registered

Classification	Image data supporting function		
Function name	Request to collect number of image display data items registered	Symbol	Res_IMG_DataCnt
Function overview	<p>This function is used to collect the free space (unit: kilobytes (KB)) and number of image display data items registered for the specified folder.</p> <p>This function supports only the download frames. * Preinstall (frame) images are not supported</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_DataCnt(folder_no , entry_info) ;		
Argument name	Type	I/O	Explanation
folder_no	unsigned short	I	Folder number (*1) RES_IMG_FOLDER_ALL: All image folders RES_IMG_FOLDER_INBOX: INBOX folder RES_IMG_FOLDER_CAM: Camera folder RES_IMG_FOLDER_USER_01: User folder 1 ~ RES_IMG_FOLDER_USER_20: User folder 20 RES_IMG_FOLDER_FRAME: Frame folder RES_IMG_FOLDER_ORGANI: Self-produced animation folder
entry_info	_RES_IMG_ENTRYINFO *	O	Pointer of free space or number of image display data items registered (*2)
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error RES_IMG_DATANOT: Data not registered
Remarks			

(*1) When RES_IMG_FOLDER_ALL is specified, the target folders are as follows:
- INBOX folder, camera folder, user folders 1 to 20, frame folder

(*2) The free space becomes invalid (0) when the self-produced animation folder is specified using folder_no.

* Because the free space is managed as all image folders, a value is returned as all the storage areas of the image display data.

Example: The values of the free space to be returned are same for a camera folder and all image folders

[Structure of Free space or number of data items registered]

```
typedef struct tagRES_IMG_ENTRYINFO
```

```
{  
    unsigned long    emptysize ;           /* [-/O] Free space (unit: kilobytes  
    ((KB))           */  
    unsigned short   rec_cnt ;             /* [-/O] Number of data items  
    registered        */  
} _RES_IMG_ENTRYINFO ;
```

11.102 Request to Collect Information about Image Display Data Paste Destination

Classification	Image data supporting function		
Function name	Request to collect information about image display data paste destination	Symbol	Res_IMG_TagetLoc_GetInfo
Function overview	<p>This function is used to collect information about current paste image display data for the specified paste destination.</p> <p>The upper-level application must support a competition, e.g., PIM locked.</p> <p>The “standby i-Appli folder” of image ID information is valid only when the standby screen is specified for the paste destination.</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_TagetLoc_GetInfo(tagetloc_get) ;		
Argument name	Type	I/O	Explanation
tagetloc_get	_RES_IMG_TAGETLOC_GETINFO *	I/O	Structure for collecting paste destination image display data (see Remarks)
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error
Remarks	<p>[Structure for collecting paste destination image display data]</p> <pre>typedef struct tagRES_IMG_TAGETLOC_GETINFO { unsigned short tagetloc ; /* [I/-] Paste destination (*1) */ unsigned short kind ; /* [-/O] Paste destination image ID information (*2) */ unsigned short img_no ; /* [-/O] Paste image No. (*3) */ } _RES_IMG_TAGETLOC_GETINFO ;</pre> <p>(*1) Specifying a paste destination: Type of tagetloc Specify one of the following values for the paste destination. Otherwise, RES_IMG_PARAM_NG (parameter NG) occurs. RES_IMG_TAGETLOC_WAITDSP: Standby screen INDEX (note 1)</p>		

RES_IMG_TAGETLOC_WAKEUPDSP: Wakeup display INDEX
 RES_IMG_TAGETLOC_TELCAL: Telephone outgoing INDEX
 RES_IMG_TAGETLOC_TELRCV: Telephone incoming INDEX
 RES_IMG_TAGETLOC_MAIL_SND: i-mail sending INDEX
 RES_IMG_TAGETLOC_MAIL_RCV: i-mail receiving INDEX
 RES_IMG_TAGETLOC_INQUIRE: Inquiry INDEX
 RES_IMG_TAGETLOC_WAITDSP_CAL: Standby calendar background
 INDEX (note 1)

(*2) Paste destination screen ID information

ID information about the image currently pasted on the specified paste destination is set.

RES_IMG_FOLDER_INBOX: INBOX FOLDER
 RES_IMG_FOLDER_CAM: Camera folder
 RES_IMG_FOLDER_USER_01: User folder 1
 ~
 RES_IMG_FOLDER_USER_20: User folder 20
 RES_IMG_FOLDER_ORGANI: Self-produced animation folder
 RES_IMG_FOLDER_PRE_INSTALL: Preinstall folder
 RES_IMG_FOLDER_I_APPLI : Standby i-Appli folder
 RES_IMG_FOLDER_WAIT_IMOTION: Standby video folder

(*3) Paste image number

Specify the following information items according to the information set in paste image ID information (*2):

Paste image ID information: INBOX FOLDER (RES_IMG_FOLDER_INBOX)

Paste image ID information: Camera folder (RES_IMG_FOLDER_CAM)

Paste image ID information: User folder 1 (RES_IMG_FOLDER_USER_01)

~

Paste image ID information: User folder 20 (RES_IMG_FOLDER_USER_20)

* index (image display data save number) of RES_IMG_HEADINFO (image display data header structure) is set.

Image display data save number: RES_IMG_NUMBER_MIN to RES_IMG_NUMBER_MAX

Paste image ID information: Preinstall folder (RES_IMG_FOLDER_PRE_INSTALL)

* For the preinstall folder, the value depends on the value in tagetloc (specified paste destination).

The following information is specified for each paste destination:

Specifying paste destination (tagetloc): Standby screen

Specifying paste destination (tagetloc): Standby calendar background image

RES_IMG_WAITDSPNOIMG: Standby screen (with no display)

Standby calendar background image (without background image)

RES_IMG_WAITDSPCAL: Standby screen (calendar)

Standby calendar background image (not used)

RES_IMG_WAITDSPANIME1: Animation 1
 RES_IMG_WAITDSPANIME2: Animation2
 RES_IMG_WAITDSPANIME3: Animation3
 RES_IMG_WAITDSPANIME4: Animation4
 RES_IMG_WAITDSPIMAGE1: Animation5
 RES_IMG_WAITDSPIMAGE2: Image 1
 RES_IMG_WAITDSPIMAGE3: Image 2
 RES_IMG_WAITDSPIMAGE4: Image 3
 RES_IMG_WAITDSPIMAGE5: Image 4
 RES_IMG_WAITDSPIMAGE6: Image 5

Specifying paste destination (tagetloc): Wakeup

RES_IMG_WAKEUP_OFF: No display
 RES_IMG_WAKEUP_MSG: Message
 RES_IMG_WAKEUP_ANIME1: Animation1
 RES_IMG_WAKEUP_ANIME2: Animation2
 RES_IMG_WAKEUP_ANIME3: Animation3
 RES_IMG_WAKEUP_ANIME4: Animation4
 RES_IMG_WAKEUP_IMAGE1: Animation5
 RES_IMG_WAKEUP_IMAGE2: Image 1
 RES_IMG_WAKEUP_IMAGE3: Image 2
 RES_IMG_WAKEUP_IMAGE4: Image 3
 RES_IMG_WAKEUP_IMAGE5: Image 4
 RES_IMG_WAKEUP_IMAGE6: Image 5

Specifying paste destination (tagetloc): telephone (outgoing or incoming), i-mail (sending or receiving), inquiry

RES_IMG_PASTE_SYSTEM1: System definition value 1
 RES_IMG_PASTE_SYSTEM2: System definition value 2
 RES_IMG_PASTE_SYSTEM3: System definition value 3

Paste image ID information: Self-produced animation folder
(RES_IMG_FOLDER_ORGANI)

RES_IMG_USERANIME_NUMBER01: Self-produced animation 1
 ~ ~
 RES_IMG_USERANIME_NUMBER20: Self-produced animation 20

Paste image ID information: Standby video folder
(RES_IMG_FOLDER_WAIT_IMOTION)

The value set in 11.20, "Setting or Referencing Image Paste Destination Specification (Msl_OpePasteSetRef)" is set here.

Paste	image	ID	information:	Standby	i-Appli	folder
(RES_IMG_FOLDER_I_APPLI_						
	RES_IMG_I_APPLI_NUMBERMIN:		Standby i-Appli image 1 (minimum value)			
			Standby i-Appli image 2 (minimum value + 1)			
	~		~			
	RES_IMG_I_APPLI_NUMBERMAX:		Standby i-Appli image 200 (minimum value + 199)			

Note 1 Perform the following steps to specify the standby calendar background:

- When the standby calendar background image is set:

<Without background image>

(1) Use the following steps to specify a calendar in the standby screen:

- Specify RES_IMG_TAGETLOC_WAITDSP (standby screen) in tagetloc (paste destination specification).
- Specify RES_IMG_FOLDER_PRE_INSTALL (preinstall) in kind (image ID information).
- Specify RES_IMG_WAITDSPCAL (calendar) in img_no (image No.).

(2) Then, use the following steps to specify without the calendar background image for the standby screen:

- Specify RES_IMG_TAGETLOC_WAITDSP_CAL (standby calendar background) in tagetloc (paste destination specification).
- Specify RES_IMG_FOLDER_PRE_INSTALL (preinstall) in kind (image ID information).
- Specify RES_IMG_WAITDSPNOIMG (without background image) in img_no (image No.).

<With background image>

(1) Use the following steps to specify a calendar in the standby screen: (* Same steps as for (without background image))

- Specify RES_IMG_TAGETLOC_WAITDSP (standby screen) in tagetloc (paste destination specification).
- Specify RES_IMG_FOLDER_PRE_INSTALL (preinstall) in kind (image ID information).
- Specify RES_IMG_WAITDSPCAL (calendar) in img_no (image No.).

* For an INBOX original image

(2) Then, use the following steps to specify with the calendar background image for the standby screen:

- Specify RES_IMG_TAGETLOC_WAITDSP_CAL (standby calendar background) in tagetloc (paste destination specification).
- Specify RES_IMG_FOLDER_INBOX (INBOX FOLDER) in kind (image ID information).
- Specify the image display data save number (note 2) (with background image) in img_no (image No.).

- When the standby calendar background image is referenced

<Without background image>

- (1) Use the above steps to check that the calendar whose standby screen paste destination is preinstall folder is specified.
 - (2) Check that the standby screen calendar paste destination is preinstall folder and without background image.
- => Then, the application determines that the calendar (without background image) is specified in the standby screen.

<With background image> * Example: Original image in INBOX

- (1) Use the above steps to check that the calendar whose standby screen paste destination is preinstall folder is specified.
 - (2) Check that the standby calendar background paste destination is image display data save number (note 2) in INBOX FOLDER is specified.
- => Then, the application determines that the calendar (with background image) is specified in the standby screen.

* This function is not used to set or reference the schedule icon because the schedule service manages the operation.

Note 2 The image display data save number indicates an image display data save number (index) stored in the image display data header information.

11.103 Request to Set Information about Image Display Data Paste Destination

Classification	Image data supporting function		
Function name	Request to set information about image display data paste destination	Symbol	Res_IMG_TagetLoc_SetInfo
Function overview	<p>This function is used to set image display data in the specified paste destination.</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_TagetLoc_SetInfo(tagetloc_set) ;		
Argument name	Type	I/O	Explanation
tagetloc_set	_RES_IMG_TAGETLOC_SETINFO *	I	Structure for setting paste destination image display data (see Remarks)
Return value	Type	I/O	Explanation
ret	Int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error RES_IMG_DATANOT: Data not registered
Remarks	<p>[Structure for setting paste destination image display data]</p> <pre>typedef struct tagRES_IMG_TAGETLOC_SETINFO { unsigned short tagetloc ; /* [I/-] Paste destination */ unsigned short folder_no ; /* [I/-] Folder having paste destination image */ unsigned short img_no ; /* [I/-] Paste image number */ } _RES_IMG_TAGETLOC_SETINFO ;</pre> <p>* See "Request to Collect Information about Image Display Data Paste Destination," for details of the values set to the above structure.</p> <p>* This function does not set the video. Use 11.20, "Setting or Referencing Image Paste Destination Specification (Msl_OpePasteSetRef)," to set the video standby.</p> <p>* Even though image display data is registered, deleted, or moved using E-2, "Request to Register Image Display Data," E-3, "Request to Erase Image Display Data," E-17, "Request to Move Image Display Data Folder," E-23, "Request to Delete Multiple Image Display Data</p>		

Items” etc., the paste destination value is not changed. The upper-level application must manage the paste destination value. Reset the value as required.

11.104 Request to Cancel Information about Image Display Data Paste Destination

Classification	Image data supporting function		
Function name	Request to cancel information about image display data paste destination	Symbol	Res_IMG_TagetLoc_Reset
Function overview	<p>This function is used to cancel the information about the paste destination of the specified original image or self-produced animation.</p> <p>* Specify the default image (see Remarks) for the paste destination cancelled.</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_TagetLoc_Reset(folder_no , img_no) ;		
Argument name	Type	I/O	Explanation
folder_no	unsigned short	I	Folder number RES_IMG_FOLDER_INBOX: INBOX FOLDER RES_IMG_FOLDER_CAM: Camera folder RES_IMG_FOLDER_USER_01: User folder 1 ~ ~ RES_IMG_FOLDER_USER_20: User folder 20 RES_IMG_FOLDER_ORGANI: Self-produced animation folder RES_IMG_FOLDER_I_APPLI: Standby i-Appli folder
img_no	unsigned short	I	Paste destination image number
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error
Remarks			

* To cancel multiple original images (delete all images), call this function more than once.

* The default values at cancellation are as follows:

No	Paste destination	Default name	Default value	Remarks
1	Standby screen	Animation 1	RES_IMG_WAITDSPANIME1	
2	Wakeup	Image 1	RES_IMG_WAKEUP_IMAGE2	
3	Telephone outgoing	Animation 1	RES_IMG_PASTE_SYSTEM1	
4	Telephone incoming	Animation 1	RES_IMG_PASTE_SYSTEM1	
5	i-mail sending	Animation 1	RES_IMG_PASTE_SYSTEM1	
6	i-mail receiving	Animation 1	RES_IMG_PASTE_SYSTEM1	
7	Inquiry	Animation 1	RES_IMG_PASTE_SYSTEM1	

See “Request to Collect Information about Image Display Data Paste Destination,” for details of the values set to the paste destination image number.

11.105 Setting or Referencing Image Display Data Paste Information (Cutout or Display Position)

Classification	Image data supporting function		
Function name	Setting or referencing image display data paste information (cutout or display position)	Symbol	Res_IMG_TagetLocInfo_SetRef
Function overview	<p>This function is used to set or reference paste information about the specified image display data save number. The information includes cutout and display position.</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_TagetLocInfo_SetRef(tagetloc_info) ;		
Argument name	Type	I/O	Explanation
tagetloc_info	_RES_IMG_TAGETLOC_INFO *	I/O	Pointer of structure for setting or referencing paste information (cutout or display position) (see Remarks)
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error
Remarks	<pre>[Structure for setting or referencing paste information (cutout or display position)] typedef struct tagRES_IMG_TAGETLOC_INFO { unsigned short mode ; /* [I/-] Operation setting (*1) */ unsigned short kind ; /* [I/-] Type (*2) */ unsigned short img_no ; /* [I/-] Image data save number (*3) */ unsigned char data ; /* [I/O] Setting information (*4) */ } _RES_IMG_TAGETLOC_INFO ;</pre> <p>(*1) Operation setting value RES_IMG_SET: Setting RES_IMG_REF: Reference</p> <p>(*2) Type setting value RES_IMG_CUT: Image cutout position</p>		

RES_IMG_POS: Image display position

(*3) Image display data save number

RES_IMG_NUMBER_MIN to RES_IMG_NUMBER_MAX

(*4) Setting information value (when operation is specified: IN information, at reference: OUT information)

RES_IMG_UPPER: Display from above

RES_IMG_CENTER: Display center

RES_IMG_LOWER: Display from beneath

11.106 Setting or Referencing User-defined Animation Information

Classification	Image data supporting function		
Function name	Setting or referencing user-defined animation information	Symbol	Res_IMG_UserAnimation_SetRef
Function overview	<p>This function is used to set or reference the frame order of the user-defined animations (self-produced animations).</p> <p>* Up to 20 frames can be set.</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_UserAnimation_SetRef(mode , anime_data) ;		
Argument name	Type	I/O	Explanation
mode	unsigned short	I	<p>Mode</p> <p>RES_IMG_SET: Sets one animation.</p> <p>RES_IMG_SET_ALL: Sets all animations.</p> <p>RES_IMG_REF : Reference one animation.</p> <p>RES_IMG_REF_ALL: References all animations.</p>
anime_data	_RES_IMG_USERANI_INFO *	I/O	Pointer of structure for setting or referencing user-defined animation information (see Remarks)
Return value	Type	I/O	Explanation
ret	int	O	<p>Processing result</p> <p>RES_IMG_OK: Normal end</p> <p>RES_IMG_NG: Abnormal end</p> <p>RES_IMG_PARAM_NG: Parameter error</p> <p>RES_IMG_NO_SPACE: Free space shortage</p>
Remarks			

[Structure for Setting or Referencing User-defined Animation Information]

```
typedef struct tagRES_IMG_USERANI_INFO
{
    unsigned char    title[RES_IMG_TITLE_MAX+1] ;    /* [I/O] Storage area for title character
string        (*1) */
    unsigned short   frame_info[RES_IMG_FRAME_MAX] ; /* [I/O] Setting data
pointer        (*2) */
    unsigned short   img_no ;                        /* [I/O] Self-produced animation registration
number        (*3) */
} _RES_IMG_USERANI_INFO ;
```

```
#define RES_IMG_FRAME_MAX    (20)        /* Maximum number of setting data
arrays        */
```

```
#define RES_IMG_ANIME_MAXCNT    (20)        /* Maximum number of self-produced
animations that can be registered        */
```

(*1) Specify a null at the end of the character string. (IN information: at setting, OUT information: at reference)

(*2) IN information: at setting, OUT information: at reference

(*3) The self-produced animation registration setting data is a setting value (IN information: valid when one animation is specified, OUT information: valid only at reference)

<Self-produced animation number>

RES_IMG_USERANIME_NUMBER01

~

RES_IMG_USERANIME_NUMBER20

Note:

- When RES_IMG_SET_ALL or RES_IMG_REF_ALL is specified in mode, the upper-level application must allocate a continuous area for the value obtained by “RES_IMG_ANIME_MAXCNT (maximum number of self-produced animations that can be registered (20) x _RES_IMG_USERANI_INFO structure.” Then, specify the starting address.

<At setting >

- For an unused frame, specify RES_IMG_USERANIME_DEL in frame_info (setting data).

<At reference>

- For an unused frame, RES_IMG_USERANIME_DEL is specified in frame_info (setting data).

11.107 Checking User-defined Animation Utilization Status

Classification	Image data supporting function																	
Function name	Checking user-defined animation utilization status	Symbol	Res_IMG_UserAnimation_Check															
Function overview	<p>This function is used to check whether the specified original image is used by the user-defined animation (self-produced animation).</p>																	
Include file	res_img.h																	
Calling sequence	int Res_IMG_UserAnimation_Check(img_no , anime_data) ;																	
Argument name	Type	I/O	Explanation															
img_no	unsigned short	I	Image display data save number															
anime_data	unsigned char *	I/O	Storage area for self-produced animation utilization status * Stores status indicating whether the specified image display data save number (img_no) is used.															
Return value	Type	I/O	Explanation															
Ret	int	O	Processing result RES_IMG_ORGUSED: Being used in the self-produced animation RES_IMG_ORGUNUSED: Not used in the self-produced animation RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error															
Remarks	<p>* When the specified image display data save number is used in the self-produced animation, RES_IMG_ORGUSED (Being used in the self-produce animation) is returned. When it is used in no self-produced animation, RES_IMG_ORGUNUSED (Not used in the self-produce animation) is returned.</p> <p>* Storage area for self-produced animation utilization status</p> <table><tr><td></td><td>anime_data</td><td></td></tr><tr><td>Self-produced animation 1</td><td>anime_data[0]</td><td>For a self-produced animation (1 to n) that is using the specified image data save number (img_no),</td></tr><tr><td>Self-produced animation 2</td><td>anime_data[1]</td><td>RES_IMG_ORGUSED (Being used) is set in anime_data.</td></tr><tr><td>Self-produced animation 3</td><td>anime_data[2]</td><td>For a self-produced animation that is not using the specified image data save number,</td></tr><tr><td>Self-produced</td><td>anime_data[3]</td><td>RES_IMG_ORGUNUSED (Not used) is set in the corresponding</td></tr></table>				anime_data		Self-produced animation 1	anime_data[0]	For a self-produced animation (1 to n) that is using the specified image data save number (img_no),	Self-produced animation 2	anime_data[1]	RES_IMG_ORGUSED (Being used) is set in anime_data.	Self-produced animation 3	anime_data[2]	For a self-produced animation that is not using the specified image data save number,	Self-produced	anime_data[3]	RES_IMG_ORGUNUSED (Not used) is set in the corresponding
	anime_data																	
Self-produced animation 1	anime_data[0]	For a self-produced animation (1 to n) that is using the specified image data save number (img_no),																
Self-produced animation 2	anime_data[1]	RES_IMG_ORGUSED (Being used) is set in anime_data.																
Self-produced animation 3	anime_data[2]	For a self-produced animation that is not using the specified image data save number,																
Self-produced	anime_data[3]	RES_IMG_ORGUNUSED (Not used) is set in the corresponding																

animation 4		anime_data (*2).
Self-produced		
animation 5	anime_data[4]	
~	~	
Self-produced		
animation n (*1)	anime_data[n-1]	

*1 n = RES_IMG_ANIME_MAXCNT (maximum number of self-produced animations: 20)

*2 For example, if the specified image data save number (img_no) is used in the self-produced animations 2 and 5, RES_IMG_ORGUSED (Being used) is set in anime_data[1] and anime_data[4]. RES_IMG_ORGUNUSED (Not used) is set in other anime_data.

11.108 Request to Create Image Display Data Folder

Classification	Image data supporting function		
Function name	Request to create image display data folder	Symbol	Res_IMG_FolderCreate
Function overview	<p>This function is used to create or rename a user folder.</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_FolderCreate(folder_info) ;		
Argument name	Type	I/O	Explanation
folder_info	_RES_IMG_FOLDERINFO *	I/O	Folder information structure pointer (see Remarks)
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error RES_IMG_NO_SPACE: No free folder
Remarks	<pre> [Folder information structure] typedef struct tagRES_IMG_FOLDERINFO { unsigned short folder_no ; /* [I/O] Folder number (*1) */ unsigned char folname[RES_IMG_FOLNAME_MAX+1] ; /* [I/-] Folder name (*2) */ unsigned long count ; /* (Not used) */ } _RES_IMG_FOLDERINFO ; #define RES_IMG_FOLNAME_MAX (20) /* Maximum number of characters registered for the folder name */ (*1) Folder number setting values * Espan lang=EN-US>IN information <Creating a user folder> RES_IMG_FOLDER_CREATE: Creates a folder. <Creating a folder with a folder number specified or renaming a folder> </pre>		

RES_IMG_FOLDER_USER_01: User folder 1

~

RES_IMG_FOLDER_USER_20: User folder 20

* Espan lang=EN-US>OUT information * This information is valid only when creating a folder and invalid when renaming a folder.

<Newly registering a folder>

RES_IMG_FOLDER_USER_01: User folder 1

~

RES_IMG_FOLDER_USER_20: User folder 20

(*2) Specify a null at the end of the character string.

Note:

- To create a folder with the folder number specified, specify RES_IMG_FOLDER_USER_01 to RES_IMG_FOLDER_USER_20 (folder number to be created) for a folder number not created.
- To rename a folder, specify RES_IMG_FOLDER_USER_01 to RES_IMG_FOLDER_USER_20 (folder number to be renamed) for a folder number already created.
- When RES_IMG_FOLDER_CREATE (Create) is requested while 20 user folders have already been created, RES_IMG_NO_SPACE is returned.

11.109 Request to Erase Image Display Data Folder

Classification	Image data supporting function		
Function name	Request to erase image display data folder	Symbol	Res_IMG_FolderErase
Function overview	<p>This function is used to erase a user folder.</p>		
Include file	Res_img.h		
Calling sequence	Int Res_IMG_FolderErase(folder_info);		
Argument name	Type	I/O	Explanation
folder_info	_RES_IMG_FOLDERINFO *	I	Folder information structure pointer (see Remarks)
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error RES_IMG_DATAFOUND: Data in folder RES_IMG_FOLDERNOT: Folder not registered
Remarks	<pre> [Folder information structure] typedef struct tagRES_IMG_FOLDERINFO { unsigned short folder_no ; /* [I/-] Folder number (*1) */ unsigned char folname[RES_IMG_FOLNAME_MAX+1] ; /* (Not used */ unsigned long count ; /* (Not used) */ } _RES_IMG_FOLDERINFO ; (*1) Folder number setting values RES_IMG_FOLDER_USER_01: User folder 1 ~ RES_IMG_FOLDER_USER_20: User folder 20 (*2) Specify a null at the end of the character string. </pre>		

Note:

- When image data is registered in the folder to which erase is specified, RES_O<G_DATAFOUND (Data in folder) is returned.
- Use E-8, "Request to Collect Number of Image Display Data Items Registered," to check whether image data is contained in the folder.

11.110 Request to Move Image Display Data Folder

Classification	Image data supporting function		
Function name	Request to move image display data folder	Symbol	Res_IMG_FolderMove
Function overview	<p>This function is used to move the folder having the specified image display data.</p>		
Include file	Res_img.h		
Calling sequence	Int Res_IMG_FolderMove(folder_move) ;		
Argument name	Type	I/O	Explanation
folder_move	_RES_IMG_FOLDERMOVE *	I	Folder move structure pointer (see Remarks)
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error RES_IMG_FOLDERNOT: Folder not registered
Remarks	<pre> [Folder move structure] typedef struct tagRES_IMG_FOLDERMOVE { unsigned short move_cnt ; /* [I/-] Number of image display data items to be moved (*1) */ unsigned short *img_no ; /* [I/-] Pointer of the save number array for the image displaydata to be moved (*2) */ unsigned short folder_no ; /* [I/-] Move destination folder number (*3) */ } _RES_IMG_FOLDERMOVE ; (*1) Range of image display data to be moved 1 ~ RES_IMG_GET_INFO_MAX(400) (*2) Allocate an area for the value obtained by (number of image display data items to be moved (move_cnt) x unsigned short area). (*3) Move destination folder number </pre>		

RES_IMG_FOLDER_INBOX: INBOX FOLDER
RES_IMG_FOLDER_CAM: Camera folder
RES_IMG_FOLDER_USER_01: User folder 1
~ ~
RES_IMG_FOLDER_USER_20: User folder 20

Note:

- The upper-level application must control an inhibited move (for example, move from INBOX FOLDER to Camera folder). The upper-level application must also control a request to move multiple images so that the images to be moved cannot have images that breach the specifications.

11.111 Request to Collect Number of Image Display Data Folders

Classification	Image data supporting function		
Function name	Request to collect number of image display data folders	Symbol	Res_IMG_Get_FolderCnt
Function overview	<p>This function is used to collect the number of user folders registered.</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_Get_FolderCnt(get_info , entry_info) ;		
Argument name	Type	I/O	Explanation
get_info	_RES_IMG_GET_INFO *	I	Collection information structure pointer (see Remarks)
entry_info	_RES_IMG_ENTRYINFO *	I/O	Pointer of structure of free space or number of data folders registered (see Remarks)
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error
Remarks	<pre> [Collection information structure] typedef struct tagRES_IMG_GET_INFO { unsigned short folder_no ; /* [I/-] Folder number (*1) */ unsigned short sortkey ; /* (Not used) */ unsigned short start_no ; /* (Not used) */ unsigned short rec_cnt ; /* (Not used) */ } _RES_IMG_GET_INFO ; [Structure of free space or number of data items registered] typedef struct tagRES_IMG_ENTRYINFO { unsigned long emptysize ; /* (Not used) */ unsigned short rec_cnt ; /* [-/O] Number of data items registered */ } _RES_IMG_ENTRYINFO ; </pre>		

(*1) Folder number setting value (Specify a folder number whose number of data items can be collected.)

RES_IMG_FOLDER_ALL: All image folders

11.112 Request to Collect Image Display Data Folder Name

Classification	Image data supporting function		
Function name	Request to collect image display data folder name	Symbol	Res_IMG_Get_FolderInfo
Function overview	<p>This function is used to collect a name of the user folder registered.</p> <p>Folder names can be collected with a folder number specified (one folder) or folder list specified (multiple folders) in the registration order, starting with the oldest.</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_Get_FolderInfo(get_info , entry_info);		
Argument name	Type	I/O	Explanation
get_info	_RES_IMG_GET_INFO *	I/O	Collection information structure pointer (see Remarks)
folder_info	_RES_IMG_FOLDERINFO *	I/O	Folder information structure pointer (see Remarks)
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error RES_IMG_DATANOT: Data not registered
Remarks	<p>[Collection information structure]</p> <pre>typedef struct tagRES_IMG_GET_INFO { unsigned short folder_no ; /* [I/-] Folder number (*1) */ unsigned short sortkey ; /* Not used */ unsigned short start_no ; /* [I/-] Collection start position (*2) */ unsigned short rec_cnt ; /* [I/O] Number of data folders collected (*3) */ } _RES_IMG_GET_INFO ;</pre> <p>[Folder information structure]</p>		

```

typedef struct tagRES_IMG_FOLDERINFO
{
    unsigned short    folder_no ;           /* [-/O] Folder number           (*4)
*/
    unsigned char     folname[RES_IMG_FOLNAME_MAX+1] ; /* [-/O] Folder
name                               */
    unsigned long     count ;               /* (Not used)                     */
} _RES_IMG_FOLDERINFO ;

```

(*1) Folder number setting values

<Specifying one folder number> * For specifying one folder number, the collection start position (start_no) and number of data folders (rec_cnt) are not used.

RES_IMG_FOLDER_USER_01: User folder 1

~ ~

RES_IMG_FOLDER_USER_20: User folder 20

<Specifying a folder list (in the registration order)>

RES_IMG_FOLDER_OLDSORT: Specifies user folders in the registration order, starting with the oldest.

(*2) A value from 1 to 20 (maximum number of user folders that can be registered) can be specified for the collection start position.

* The value set becomes valid only when RES_IMG_FOLDER_OLDSORT is specified for the folder numbers. Otherwise, this member is invalid.

* Information is collected from the nth folder from the start, among the folders in the registration order, starting with the oldest.

(*3) A value from 1 to 20 (maximum number of user folders that can be registered) can be specified for the number of data items collected.

IN information: Specifies the number of data items to be collected.

OUT information: Specifies the number of data items collected.

(*4) Folder number values

RES_IMG_FOLDER_USER_01: User folder 1

~

RES_IMG_FOLDER_USER_20: User folder 20

Note:

- For the folder information structure (folder_info), allocate an area for the size specified in rec_cnt (number of data items collected) of get_info (collection information structure). When the user folder number is specified, allocate an area for one folder.

11.113 Request to Collect Name of Folder that Registers Image Display Data

Classification	Image data supporting function		
Function name	Request to collect name of folder that registers image display data	Symbol	Res_IMG_Get_FolderNo
Function overview	<p>This function is used to collect the name of the storage destination folder according to the specified image display data save number and self-produced animation ID.</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_Get_FolderNo(img_no);		
Argument name	Type	I/O	Explanation
img_no	unsigned short	I	<p>Image display data save number/self-produced animation ID</p> <p>* When the self-produced animation ID is specified, specify the following value:</p> <p><self-produced animation ID></p> <p>RES_IMG_USERANIME_NUMBER01: Self-produced animation 1</p> <p>~</p> <p>RES_IMG_USERANIME_NUMBER20: Self-produced animation 20</p>
Return value	Type	I/O	Explanation
ret	int	O	<p>Processing result</p> <p>RES_IMG_FOLDER_INBOX: INBOX FOLDER</p> <p>RES_IMG_FOLDER_CAM: Camera folder</p> <p>RES_IMG_FOLDER_USER_01: User folder 1</p> <p>~</p> <p>RES_IMG_FOLDER_USER_20: User folder 20</p> <p>RES_IMG_FOLDER_FRAME: Frame folder</p> <p>RES_IMG_FOLDER_ORGANI: Self-produced animation folder</p> <p><Other return values></p> <p>RES_IMG_NG: Abnormal end</p>

			RES_IMG_DATANOT: Data not registered RES_IMG_PARAM_NG: Parameter error
Remarks	<p>* When the specified image display data save number is not registered, RES_IMG_DATANOT (Data not registered) is returned.</p>		

11.114 Setting or Referencing Display Switching Information

Classification	Image data supporting function		
Function name	Setting or referencing display switching information	Symbol	Res_IMG_DspChg_SetRef
Function overview	<p>This function is used to set or reference the display switching sort information for a volatile memory (Msl_OpeSetRef).</p> <p>* This function also sets or references the soft conditions when the image list screen is displayed.</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_DspChg_SetRef(mode, info);		
Argument name	Type	I/O	Explanation
mode	unsigned short	I	Specifies mode. RES_IMG_SORTKEY_SET: Sets a sort display order. RES_IMG_SORTKEY_REF: References a soft display order.
info	unsigned short *	I/O	Switching display type (*1) RES_IMG_TIME_ASC: File date, starting with the oldest RES_IMG_TIME_DESC: File date, starting with the newest RES_IMG_TITLE_ASC: Title name, ascending order RES_IMG_TITLE_DESC: Title name, descending order RES_IMG_SIZE_ASC: Image display data size, starting with the smallest RES_IMG_SIZE_DESC: Image display data size, starting with the greatest RES_IMG_GET_METHOD: File collection order
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error
Remarks			

(*1) For the switching display type, at setting: IINPUT information/at reference: OUTPUT information.

11.115 Request to Collect Image Folder Image ID

Classification	Image data supporting function		
Function name	Request to collect image folder image ID	Symbol	Res_IMG_GetImgNum
Function overview	<p>This function is used to collect the image display data save numbers of all images registered in the specified folder.</p> <p>* This function also creates a list (ascending order) of save numbers for the image display data registered in the specified folder. It writes the numbers to imgnolist (upper-level specification area).</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_GetImgNum(folder_no, imgnolist);		
Argument name	Type	I/O	Explanation
folder_no	unsigned short	I	Folder number to be collected RES_IMG_FOLDER_ALL: All image folders RES_IMG_FOLDER_INBOX: INBOX FOLDER RES_IMG_FOLDER_CAM: Camera folder RES_IMG_FOLDER_USER_01: User folder 1 ~ RES_IMG_FOLDER_USER_20: User folder 20 RES_IMG_FOLDER_FRAME: Frame folder
imgnolist	unsigned short *	O	List of Image display data save numbers (*1) Pointer for storing image display data save number to be collected
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_IMG_OK: Normal end RES_IMG_NG: Abnormal end RES_IMG_PARAM_NG: Parameter error
Remarks			

(*1)

- The upper-level application must allocate an area for the number of folders by using E-8, "Request to Collect Number of Image Display Data Items Registered."
- When RES_IMG_FOLDER_ALL (all image folders) is specified, all image display data save numbers in the following folder are collected.
INBOX FOLDER, Camera folder, user folders 1 to 20, frame folder

11.116 Request to Delete Multiple Image Display Data Items

Classification	Image data supporting function		
Function name	Request to delete multiple image display data items	Symbol	Res_IMG_DataErase_List
Function overview	<p>This function is used to delete image display data for the specified number.</p>		
Include file	res_img.h		
Calling sequence	int Res_IMG_DataErase_List(imgnolist);		
Argument name	Type	I/O	Explanation
imgnolist	unsigned short *	I	<p>List of image display data deletion numbers (*1)</p> <p>Pointer for storing the image display data save number to be deleted</p>
Return value	Type	I/O	Explanation
ret	int	O	<p>Processing result</p> <p>RES_IMG_OK: Normal end</p> <p>RES_IMG_NG: Abnormal end</p> <p>RES_IMG_PARAM_NG: Parameter error</p>
Remarks	<p>(*1)</p> <ul style="list-style-type: none"> - Specify 0 for the last data on the list of image display data deletion numbers. - The maximum number of image display data deletion numbers on the list must be RES_IMG_GET_INFO_MAX (400). If 0 is set before RES_IMG_GET_INFO_MAX (400), the numbers until 0 are valid. <p>* The following values can be specified on the list of image display data deletion numbers:</p>		

Image display data save number: RES_IMG_NUMBER_MIN to
RES_IMG_NUMBER_MAX

11.117 Request to Collect Icon Information

Classification	Icon data supporting function		
Function name	Request to collect icon information	Symbol	Res_Icon_GetImageInfo
Function overview	<p>This function collects information about the specified icon and size of the icon.</p>		
Include file	res_icon.h		
Calling sequence	int Res_Icon_GetImageInfo(icon_id, icon_info);		
Argument name	Type	I/O	Explanation
icon_id	Int	I	Icon ID (*1)
icon_info	_RES_ICON_GETIMAGEINFO *	O	Icon information data structure pointer (*2)
Return value	Type	I/O	Explanation
ret	Int	O	<p>Processing result</p> <p>RES_ICON_OK: Normal end</p> <p>RES_ICON_NG: Abnormal end</p> <p>RES_ICON_PARAM_NG: Parameter error</p>
Remarks	<pre> *2 [Icon information data structure] typedef struct tagRES_ICON_GETIMAGEINFO { unsigned int width; /* Icon width */ unsigned int height; /* Icon height */ unsigned int col_size; /* Number of icon colors */ unsigned int data_size; /* Icon data size */ } _RES_ICON_GETIMAGEINFO ; Information set in col_size above: #define RES_ICON_256COLOR /* 256-color image */ #define RES_ICON_GIF /* GIF image */ #define RES_ICON_JPG /* JPEG image */ #define RES_ICON_SWF /* SWF image */ #define RES_ICON_SWF /* SWF image */ #define RES_ICON_64KCOLOR /* 64K color bit map image */ </pre>		

Note:

To collect an icon image, use F-4, "Request to Collect Icon Image," according to the icon data size information collected using this function.

11.118 Request to Collect Icon Image

Classification	Icon data supporting function		
Function name	Request to collect icon image	Symbol	Res_Icon_GetImageData
Function overview	<p>This function is used to collect an icon image of the specified icon type.</p>		
Include file	res_icon.h		
Calling sequence	int Res_Icon_GetImageData(icon_id, image) ;		
Argument name	Type	I/O	Explanation
icon_id	int	I	Icon ID (*1)
image	void *	O	Icon image storage pointer (*2)
Return value	Type	I/O	Explanation
ret	int	O	Processing result RES_ICON_OK: Normal end RES_ICON_NG: Abnormal end RES_ICON_PARAM_NG: Parameter error
Remarks	<p>*2 Allocate an area for the icon data size (value specified in data_size) collected using F-3, "Request to Collect Icon Information," for the icon image area. The upper-level application must allocate this area.</p>		